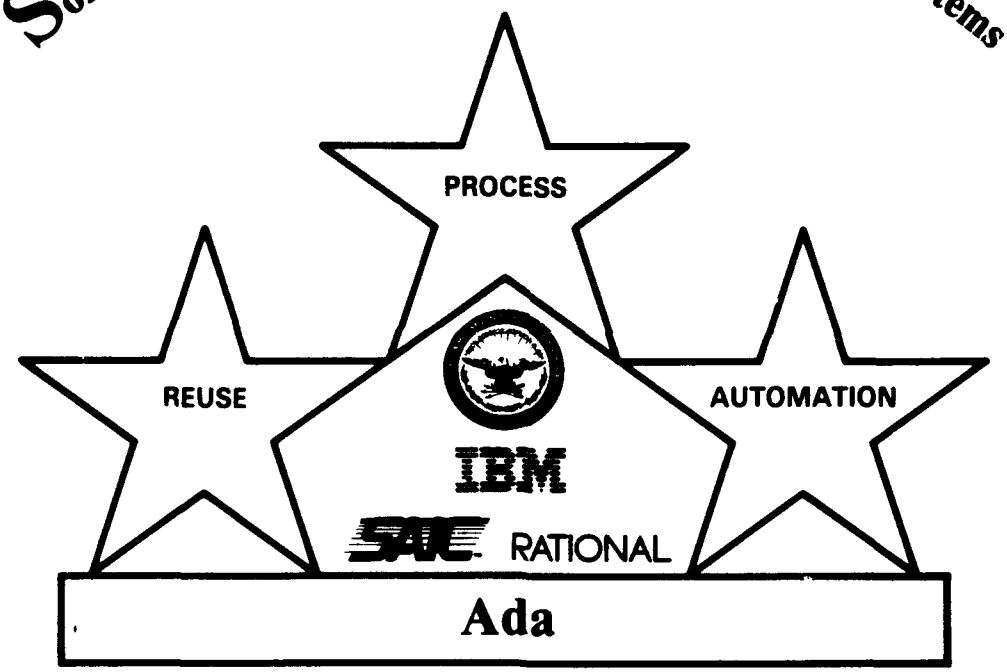
Document No. 1240-001
15 January 1990

(2)

Software - First Life Cycle Final Definition

for the

Software Technology for Adaptable Reliable Systems



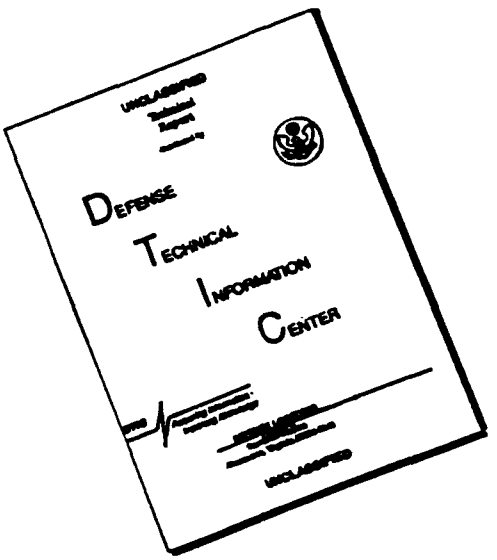
Contract No. F19628-88-D-0032

Task IQ15 — Design Process Plan

CDRL Sequence No. 1240

15 January 1990

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	Janury 15, 1990	Final

4. TITLE AND SUBTITLE
Software-First Life Cycle Final Definition

5. FUNDING NUMBERS
C: F19628-88-D-0032

6. AUTHOR(S)
M. Blumberg, M. C. Ward

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
IBM Federal Sector Division
800 N. Frederick Avenue
Gaithersburg, MD 20879

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Electronic Systems Division
Air Force Systems Command, USAF
Hanscom AFB, MA 01731-5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
CDRL Sequence No. 1240

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This document formally defines each phase of the Software-First Life Cycle (SFLC). The SFLC provides a ramatically different approach to systems development by integrating the use of rapid prototyping, software reuse, concurrent engineering and other emerging technologies. This approach, which takes advantage of each of the included technologies, has the potential for substantial improvements in productivity while increasing the quality and reusability of the developed system.

14. SUBJECT TERMS
STARS, Software-First Life Cycle, rapid prototyping, software reuse

15. NUMBER OF PAGES
90

16. PRICE CODE

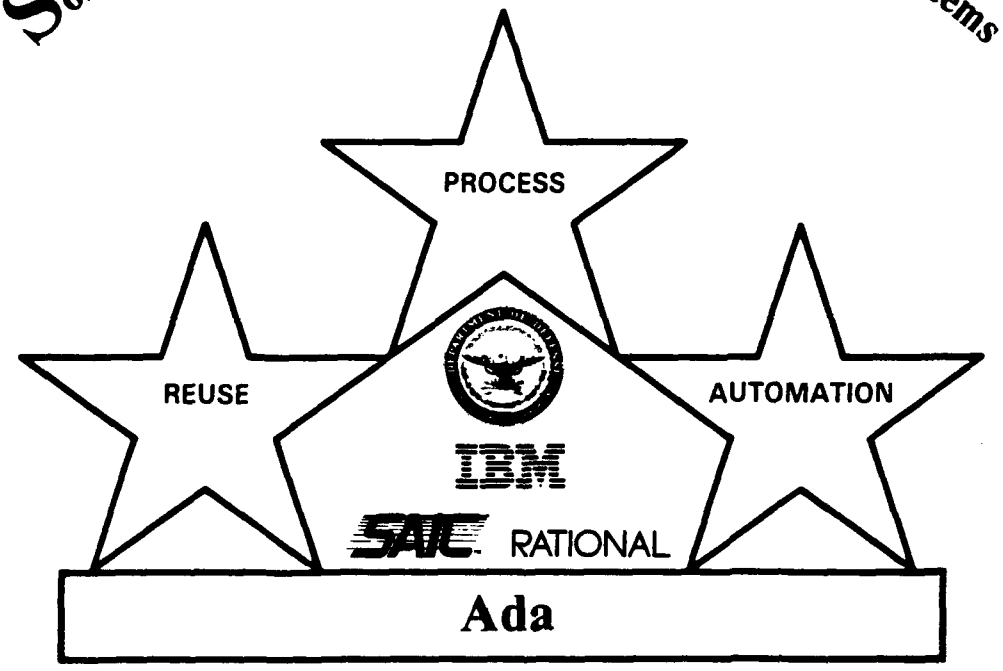
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

Document No. 1240-001
15 January 1990

Software - First Life Cycle
Final Definition

for the

Software Technology for Adaptable Reliable Systems



Contract No. F19628-88-D-0032

Task IQ15 — Design Process Plan

CDRL Sequence No. 1240

15 January 1990

Software - First Life Cycle
Final Definition
for the
Software Technology for Adaptable, Reliable Systems
(STARS) Program

Contract No: F19628-88-D-0032

Task IQ15 — Design

CDRL Sequence No. 1240

15 January 1990

Prepared for:

Electronic Systems Division
Air Force Systems Command, USAF
Hanscom AFB, MA 01731-5000

Prepared by:

IBM Systems Integration Division
800 North Frederick Avenue
Gaithersburg, MD. 20879

SOFTWARE-FIRST LIFE CYCLE FINAL DEFINITION

Document Number CDRL Sequence No. 01240

January 5th, 1990

IBM System Integration Division

800 N. Frederick Ave.
Gaithersburg, Maryland 20879

Abstract

This document, IBM Contract Data Requirements List (CDRL) Sequence Number 1240, describes the final definition of a Software-First Life Cycle (SFLC), developed under the Q and R increments of the STARS program. The SFLC provides a dramatically different approach to systems development by integrating the use of rapid prototyping, reusable components, concurrent engineering, and other new and emerging technologies. This approach, which takes advantage of the benefits of each of the included technologies, has the potential for a substantial improvement in productivity, while increasing the quality and reusability of the developed system.

The SFLC consists of 5 phases: Preliminary System Analysis, System Architecture, Software Growing, Productization and Production, and System Operation and Support. Each of these phases are formally defined using the ETVX (Entry Criteria, Task, Validation, and Exit Criteria) model. The major input, output and intermediate work products are also identified and defined.

This document results from the work of the IBM Q15 team of Maurice Blumberg and Mary Catherine Ward.

Table of Contents

List of Illustrations

Introduction

Goals

The goal of the STARS program is to develop a system development process and the new technologies needed to support it. These will "provide a basis for the development of very large, complex (and in particular, the so-called embedded, real time) systems on a predictable time scale far shorter and for a cost far less than is possible with today's software technology and acquisition process, and with a quality that far exceeds the norm." (Gree89) The system development process is based on a "software-first" approach for new system acquisition, as defined in Attachment 4 to AF Regulation 800-14 (STAR87). The definition of the Software First Life Cycle (SFLC), descibed in this paper, is consistent with the 800-14 definition and is intended to be the software-first process which meets the STARS goals.

In developing a definition of the Software First Life Cycle (SFLC), our goals were to define a life cycle process that:

- Takes full advantage of the lessons learned concerning existing life cycle processes. (See "Analysis of the Existing System Development Process" on page 3.)
- Takes full advantage of the new and emerging technologies. (See "Fundamental Concepts of the SFLC" on page 5.)
- Is "executable", i.e., provides enough detail and structure so that it can be implemented and followed on a real project. (See "High Level View of the SFLC" on page 8, "SFLC Process Model" on page 11 and "Major Work Products of the SFLC Phases" on page 52.)
- Is flexible enough to be applied in the near term while the new technologies are still maturing, and in the long term when the new technologies have matured. (See "High Level View of the SFLC" on page 8, "SFLC Process Model" on page 11 and "Major Work Products of the SFLC Phases" on page 52.)
- Is flexible enough to be applied to precedented systems i.e., those systems with well-defined technologies and/or previous implementations, and unprecedented systems, i.e., those systems requiring new technologies and/or with no previous implementations. (See "High Level View of the SFLC" on page 8, "SFLC Process Model" on page 11 and "Major Work Products of the SFLC Phases" on page 52.)

Approach

In developing the SFLC defined in this document, the experiences of several projects were studied, many current research efforts were evaluated, and various technical exchanges were conducted internal and external to IBM. Special emphasis was devoted to lessons learned from projects using traditional waterfall life cycles and projects with successful experiences in using new technologies such as rapid prototyping and reusability (see "Appendix B. Major Site Visits and Meetings" on page 79 and "Appendix C. Bibliography" on page 81).

Relationship to Other STARS Tasks

The SFLC, in many ways, is related to all the STARS tasks since it is the process under which all the STARS technologies will be integrated. It is most directly related to IR66, Software-First CDRLs, which will define the data item descriptions (DIDs) for the work products, as initially identified and defined in this document. In addition, IR20, Process/Environment, will define application blueprints which can be used, especially in the Preliminary System Analysis and System Architecture phases, to develop the approach to defining and implementing system prototypes and evolving them into a productized system. IQM15, Software First Systems Analysis, will define a methodology that can be used to perform the domain analysis and prototype definition and design required in the Preliminary System Analysis and System Architecture phases. IR40, Repository Integration, will provide a repository of reusable Ada components which are needed to support "rapid" prototyping and the evolution of system prototypes into a productized system.

Structure of Document

The first section, "Analysis of the Existing System Development Process" on page 3, describes some of the key problems with existing system life cycles. The section, "Fundamental Concepts of the SFLC" on page 5, summarizes the technologies which play a significant part in the SFLC. The section, "High Level View of the SFLC" on page 8, provides an overview of the SFLC phases. The section, "SFLC Process Model" on page 11, provides a formal definition of the SFLC. It describes the entrance criteria, tasks, validations and exit criteria for each of the SFLC phases. The section, "Major Work Products of the SFLC Phases" on page 52, identifies the major input, output, and intermediate work products for each SFLC phase. It gives a narrative description of each phase, and the key work products of the phase. The final section, "Summary of Work Products" on page 59, presents a summary description of the work products in table format. "Appendix A. SFLC Work Product Descriptions" on page 64 provides definitions of the major work products in alphabetical order. The last two appendices, "Appendix B. Major Site Visits and Meetings" on page 79 and "Appendix C. Bibliography" on page 81, provide a list of the key projects and people consulted and a set of reference papers/documents.

Analysis of the Existing System Development Process

The waterfall life cycle model (Royc70) is the best known and most widely used model today for the development of software. In many ways it is "the best we have", providing a well-defined and disciplined approach to systems development. Before defining a new process, we realized it was important to first examine the existing waterfall life cycle, especially in terms of lessons learned. Jim Moore, the STARS IBM System Architect, having researched this issue, identified the following key areas of concern.

Early Requirements Freezing

The waterfall life cycle starts with the formulation of a set of requirements for the software system. Once these requirements are "frozen", changes may still be made, but to maintain stability only the most needed changes are accepted. This early freezing of requirements, years prior to the actual delivery of the system, has become one of the main concerns about the use of the waterfall life cycle. As the actual requirements continue to evolve and change, early freezing of the formal requirements provides little opportunity to determine whether the system as delivered will meet the users' needs at the time of delivery. Also, technologies which emerge while the system is under development are often not exploited because the requirements have already been frozen.

Early Selection of Hardware

Typically, the computing hardware for the system is selected shortly after the requirements are frozen, possibly years before the delivery of the system. This early selection often means that the hardware will be obsolete by the time the system is fielded, because the capability of commercially available computing equipment roughly doubles every three years.

Limited User Involvement

The waterfall process tends to isolate the actual end user from the system during development. This happens because the intermediate products of the waterfall life cycle are documents, not products. Giving the user documents to evaluate a system is akin to shopping for a car and seeing only the owner's manual. The end user does not get to "test drive" the system until the last phase of the waterfall process.

Incremental Quality Control

As practiced by the DoD, software development proceeds through seven distinct phases. The basic quality control technique is to terminate each phase with a series of reviews which evaluates the documents produced by the current phase and compares them with the documents produced by the prior phase to ensure that they reconcile. This is called an "incremental" quality control technique

because comparison is primarily made to the result of the prior phase rather than to the original requirements of the system.

Since a contractor is committed to following a single development path, this approach creates a high risk in developing a system which meets its requirements. To illustrate this, consider a simple mathematical model that was applied to a hypothetical software system of 20 components proceeding through the seven phases of the DoD software development process. The probability of success for each individual component in each development phase was set high, at 99 percent. Yet, the aggregate probability of success turns out to be low, less than 25 percent, because the incremental quality control mechanism has the effect of compounding the risk of error at each stage. Remedying such potential failures requires rework of the system or redevelopment.

Fragmented Requirements

In many cases, an executable version of the entire system usually does not exist until system integration and test, the last phase of the waterfall process. (Documents are used as the system integration mechanism until then.) This implies that the focus of the system development effort is on the development, test and evaluation of the system components as individual entities. While each component may meet its requirements, without integration with each system component it can not be determined whether the system requirements are being met. In addition, there is no opportunity for the end user to evaluate the system requirements until all component development has completed.

Fundamental Concepts of the SFLC

To formulate the concepts that underlie our Software-First Life Cycle, we evaluated new and emerging technologies that have the potential for significantly improving the system development life cycle. This included studying the current literature and industry experiences. The results, combined with the lessons learned from the current system development process analysis, are the basis for the following fundamental concepts we have incorporated into our Software-First Life Cycle.

Prototyping

The ability to quickly build a partially complete working prototype of a system and progress it to a fully productized system is critical to the SFLC. Accurate system capabilities can be formulated using prototyping because prototypes give the customer team a more tangible work product to evaluate than the traditional requirements specification (Luqi89). In addition, prototyping allows for the evaluation of design alternatives, performance, and other issues early in the life cycle.

Reuse

Reuse is valuable in all phases of the system development life cycle. Reusable components should not be limited to code. There are many valuable work products that should be considered for reuse such as lessons learned, standards, application blueprints, domain analyses. The reader is referred to (IBM1540) and (IBM380) for a complete discussion on reusability.

Concurrent Engineering

Traditionally, the disciplines involved in system development interact in a sequential fashion, with one discipline usually responsible for a single step. The goal of concurrent engineering is to improve the development process by integrating activities and disciplines which were traditionally sequential. These activities and disciplines would then run concurrently, interacting heavily.

For example, traditionally in large systems development the engineering phase occurs after the research phase with little interaction. An application of concurrent engineering might be to begin systems engineering and to identify known and new technology areas. Research in the new areas can run concurrently with the engineering of known capabilities. Process tailoring and mutidisciplinary teams are two important aspects of concurrent engineering which are also elaborated in the paragraphs that follow. (Char88)

Process Tailoring

Concurrent engineering allows for the customizing of the development process concurrently with the engineering of the system. Every system is unique. This uniqueness should also be favorably exploited in terms of the life cycle process. The process should be tailored to the specific system under development. Unnecessary tasks and work products should be removed and essential unique

tasks should be added. Tasks which can be done concurrently instead of serially should be done concurrently. Who should be involved (customer team, planner, developer, etc.) in the process at what time may also change for each project. By tailoring the process, it is streamlined for a particular development effort with the intent of shortening the life cycle, decreasing costs, increasing quality and most importantly meeting the customers requirements. (Char88)

Most of the tailoring occurs at the beginning of a project, but throughout the project the process should be monitored and adapted to the evolving system development efforts.

Open Communication with Customer Team

Communication between the contractor and customer team (contracting agency and end users) must remain open throughout the entire life cycle. There should be a continuous effort from both sides to ensure that the system being developed meets the customer team's needs.

Often, the customer team has only a general idea of what the system should do or look like, and very often the needs and expertise of the customer team change as the system evolves. By working with the customer team from the beginning, the contractor can refine and evolve the system with their guidance and approval. The customer team should be encouraged to participate in check point activities such as prototype evaluations and reviews and in analysis tasks where trade-offs will be discussed and decisions made.

Involving the customer team promotes team spirit between the contractor and customer. Instead of standing on the sidelines waiting for delivery, the customer team takes an active role in creating the system, building enthusiasm, cooperation and a sense of ownership along the way.

Multidisciplinary Teams

One of the problems with the current system development practices has been the acquisition of hardware before the software component is analyzed. Another problem is that the customer team is not always involved with the project until the very end. These problems arise because the right people are not brought into the process at the right time.

The manufacturing industry promotes the use of a multidisciplinary team as one of the principles of concurrent engineering. At least one member from each of the major process areas (for system development this might include hardware, software, quality, and planning) is represented on the team. The team analyzes the system from each functional perspective and is responsible for tailoring the process to ensure that the right things are done at the right time and that the right people are included at the right time. (Char88) We propose that this includes the customer team.

While manufacturing uses only one team at the process level, extending this concept to various pieces of system development (such as the definition of user interfaces) may be worthwhile. These scaled down teams could focus on particular issues spanning multiple disciplines.

Domain Analysis

Domain analysis captures the essential characteristics of a defined domain. Objects, operations and characteristics common to all systems in the domain are identified, classified, and generalized. The result is a domain model, "transcending" specific applications. This model provides "expert" knowledge to the system analysis phase of system development. This knowledge aids decisions concerning reuse opportunities, component generation, prototype assembly, etc. (Prie87)

Hardware Independence

Another problem with current system development practices is that many decisions are made very early in the life cycle. Many of these decisions involve environment issues, such as which hardware or operating system to use. If made too soon, these decisions can cause the system to be obsolete by the time the software is completed.

Part of the solution to this problem is to make all software as portable as possible. If the software is portable, the hardware and environment decisions can be postponed until as late as possible in the life cycle. This allows the system to incorporate the latest developments in hardware and operating systems. If the software that needs to be dependent on the environment is kept to a minimum and isolated, effort to port the system is minimal.

High Level View of the SFLC

Figure 1 on page 10 presents a high level view of the Software First Life Cycle. The SFLC consists of 5 phases: Preliminary System Analysis, System Architecture, Software Growing, Productization and Production, and System Operation and Support. The phases are similar to those defined in the draft standard, "Software-First Development Standard for Systems-in-the-Large (Draft)" (STAR87). The SFLC, as defined in this document, is a refinement of the draft standard into an "executable" life cycle which integrates such technologies as concurrent engineering, rapid prototyping, and reusability. By combining and integrating these technologies into a well-defined and highly concurrent process, the SFLC has the potential for a substantial improvement in productivity while at the same time, increase the quality of the system.

The SFLC, unlike more traditional waterfall life cycles, is not sequential and is not specification or document driven. Evaluation of system capabilities and trade-offs do not occur against a set of documents, but rather against incremental prototypes which have been "rapidly" developed from reusable components. This is not to say that documents do not play an important role, just a different role, i.e., they capture the results and record the direction of system development instead of determining it. System requirements and design documents are definitized after a full-capability prototype is developed rather than at the start of a project when system requirements are not well-understood and are subject to change. Thus, the "excessive paper" production/maintenance, required of the system developer, and the "excessive paper" review, required of the customer is eliminated.

The first phase of the life cycle is Preliminary Systems Analysis, which begins with the receipt of the desired operational capability for the system to be developed. The main purpose of this phase is to understand the system and users' requirements well enough to enable prototyping to begin in the next phase. The workproducts produced by this phase provide only a partial description of the system, and its operational capabilities, and are intended to provide sufficient detail to allow development of the initial system prototype. It is also in this phase that the process definition is tailored to the unique requirements of the project, the development environment is built, and long lead time new technologies are identified to allow the required research and development efforts to be initiated.

The System Architecture phase is the driving phase of the life cycle. In this phase a full capability system prototype is evolved from a "rapidly" developed initial system prototype in order to address technological and programmatic risks early in the system life cycle and to allow early, effective involvement of the customer team in evaluating and refining system capabilities. The initial system prototype, and subsequent incremental system prototype builds, are assembled, to the greatest extent possible, from reusable components in the repository. It is during this phase that architecture trade analyses (e.g., hardware/software, performance/size) are performed. Alternative approaches are evaluated, as required, by building prototypes reflecting the alternative approach. When new software components or a new technology needs to be developed, requests are made to the Software Growing phase. Thus, as part of the process of evolving a full capability system prototype, incremental system prototypes are developed concurrently with new software components and new technologies. In addition, incremental system prototypes addressing alternative approaches of the same capabilitiy or addressing distinct capabilities are developed in parallel. These prototypes allow early involvement of the customer team, including planned users, in evaluating the system as it is being developed.

Within Software Growing new software components are developed, also using prototyping. Full capability component prototypes are productized and stored in the repository. Incremental component prototypes can also be stored in the repository to allow early use of software components in the assembly of a system prototype. It is also in this phase that new technologies, as needed, are researched and developed. When a new technology has been sufficiently defined, a new software component will be developed, as described previously, and stored in the repository.

Upon completion of the development of a full capability system prototype in the System Architecture phase, the prototype is productized in the Productization and Production phase. In this phase the system prototype, "a functionally executable software and hardware system (STAR87)", is converted into a productized system, "a fully defined, specified, developed, documented, tested, delivered and supported production quality software/hardware system (STAR87)".

Finally, in the System Operation and Support phase, the productized system is operated and maintained at the customer site. New operational requirements are identified for the next version of the system and these are input to the Systems Analysis phase. The SFLC is then repeated, as required, for a new version of the operational system.

Two major reviews, the Pre-Prototyping Review and the Pre-Productization Review are also shown in Figure 1. The Pre-Prototyping Review is performed after completion of the System Analysis phase to ensure that the system developer has a sufficient understanding of the system and users' requirements, to build a prototype. The review also ensures an understanding between the customer team and the system developer of the life cycle process tailoring, milestones, and deliverables to be followed by the project. The information that is reviewed, although not a complete definition, nevertheless, does provide a very concise definition of the system being developed and the capabilities of the initial prototype. The Pre-Productization Review is performed after completion of the System Architecture phase to ensure that the customer and the the system developer agree on the tasks necessary to productize the full capability prototype into an operational system.

Note:

In order to maintain diagram simplicity, feedback loops from a review or phase back to a previous phase are omitted from Figure 1. These feedback loops will occur when it is determined that further work is required in that phase.

Figure 1 also shows interaction with the repository in all phases of the life cycle, to indicate that more than just software is stored in the repository. In addition to software, such valuable information as lessons learned, results of domain analyses, and specifications will be stored and retrieved from the repository.

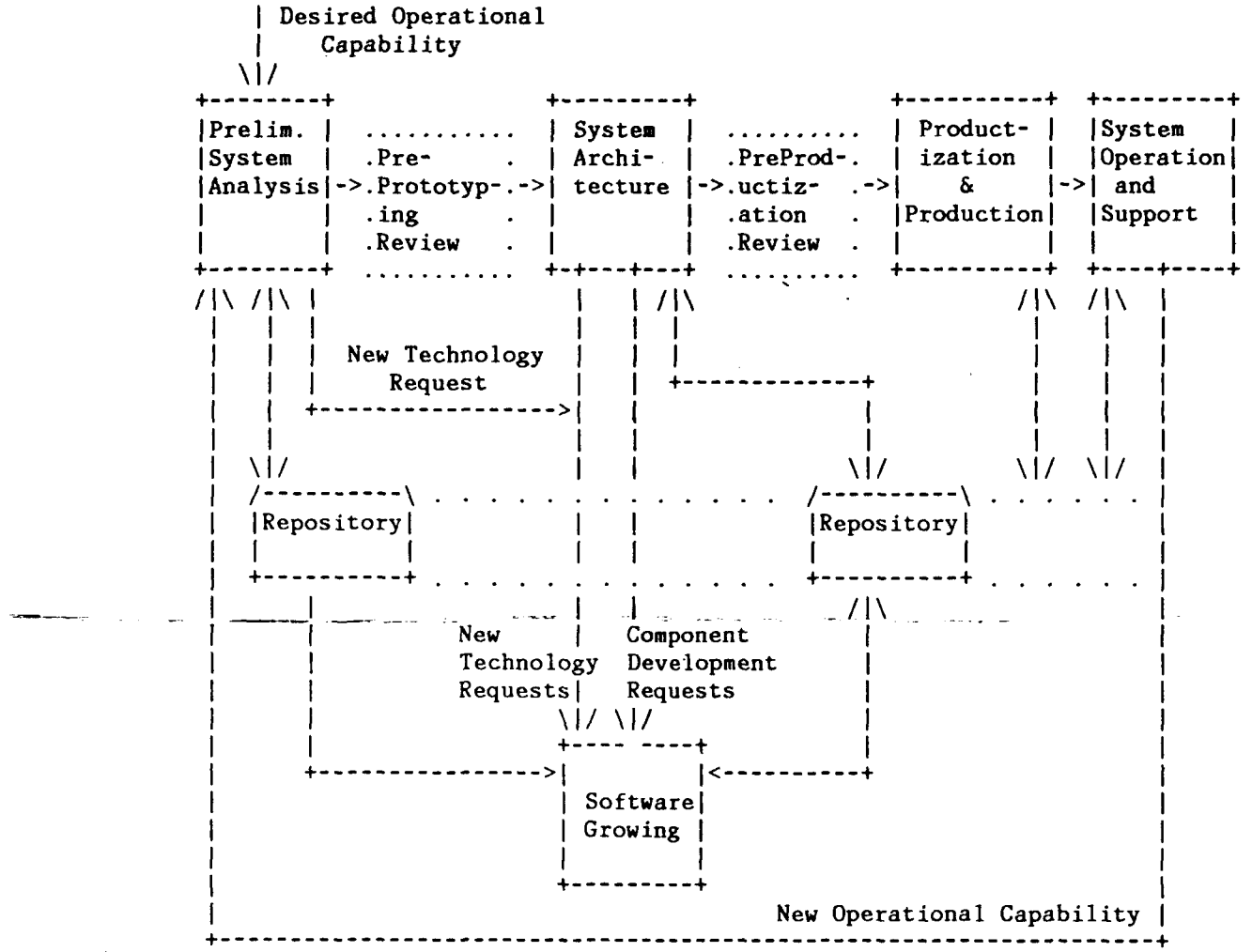


Figure 1. Software First Life Cycle - High Level View

SFLC Process Model

The purpose of this section is to present a formal definition, i.e., a process model, of the SFLC. The process model provides the framework for describing the required tasks and controls, performed within each phase of the SFLC, at a level of detail that makes the life cycle "executable", i.e., it can be used by "real" projects to develop "real" systems. The process model emphasizes what is to be accomplished in each phase, not how it is to be done and, thus, does not contain details of specific methodologies or tools. This allows it it to be applicable to a broad range of system solutions and flexible enough to be "tailorable" to multiple methodologies and tools.

The mechanism chosen for defining the SFLC process model. is the ETVX (entry criteria, task, validation, and exit criteria) model. This model has been used by various organization, including IBM, to define development processes (e.g., Rad85 and Hum89). The EVTX model can be applied at multiple levels of detail, beginning at the top level and continuing down to as fine a level of detail as is required to fully define a process.

ETVX models are described in this section for each phase of the SFLC. In addition, for the Preliminary System Analysis, System Architecture, and Software Growing phases, ETVX models are provided for each of the tasks within these phases. In future work, ETVX models will be developed for tasks within the other phases (Productization and Production phase and System Operation and Support phase) and for each subtask within a task (and possibly at lower levels).

For each SFLC phase, the ETVX model defines the following information:

- the set of inputs that are needed to perform that phase. Some of these inputs are essential, while others are used as reference material.
- entry criteria that should be satisfied before beginning that phase. The entry criteria does not mean that inputs must be complete before the phase begins, unless it is explicitly stated as part of the entry criteria.
- the tasks that describe what is to be accomplished.
- a set of validations that are used to verify the quality and completeness of the outputs produced by the phase.
- exit criteria that should be satisfied before the phase is viewed as complete.
- the set of outputs that are produced by the phase.

For the task level ETVX models, the information described above is provided for each task and includes descriptions of each subtask.

Although the SFLC is segmented into phases and tasks which are presented sequentially in the subsections below, much of the actual work occurs concurrently in various degrees across the SFLC, e.g., when multiple system and component prototypes are being developed. Thus, the ETVX models are not intended to to imply that all tasks in a one phase must wait for completion of all tasks in another phase. However, in order for a phase to have all tasks exit from it fully, all exit criteria must be satisfied at some point in the development life cycle for a given system.

The SFLC phases, and their associated tasks, are listed below and are described in the subsections that follow.

- Preliminary System Analysis Phase
  - Analyze Desired Operational Capabilities
  - Develop Plans
  - Perform Preliminary Application Blueprinting
  - Assemble Environment
- System Architecture Phase
  - Assemble System Prototype
  - Evaluate System Prototype
  - Perform Expanded System Analysis
  - Perform Pre-Productization Analysis
- Software Growing Phase
  - Develop Component Prototype(s)
  - Evaluate Component Prototype
  - Productize Component Prototype
  - Perform Component Test and Acceptance
  - Develop New Technology
- Productization and Production Phase
  - Productize Full-Capability System Prototype
  - Test Productized System
  - Run Acceptance Test of System
  - Deliver System(s)
- System Operation and Support Phase
  - Operate System
  - Maintain System
  - Identify New Operational Capabilites

Preliminary System Analysis Process Model

The process model for the Preliminary System Analysis phase is shown in Figure 3 on page 15.

The entry criteria for this activity is the receipt of project funding. The inputs, as shown in the figure, are provided by the customer team.

The tasks performed during the Preliminary System Analysis Phase are:

- Analyze Desired Operational Capabilities
- Develop Plans
- Perform Preliminary Application Blueprinting and
- Assemble Environment.

The relationship between Preliminary System Analysis tasks are shown in the figure below.

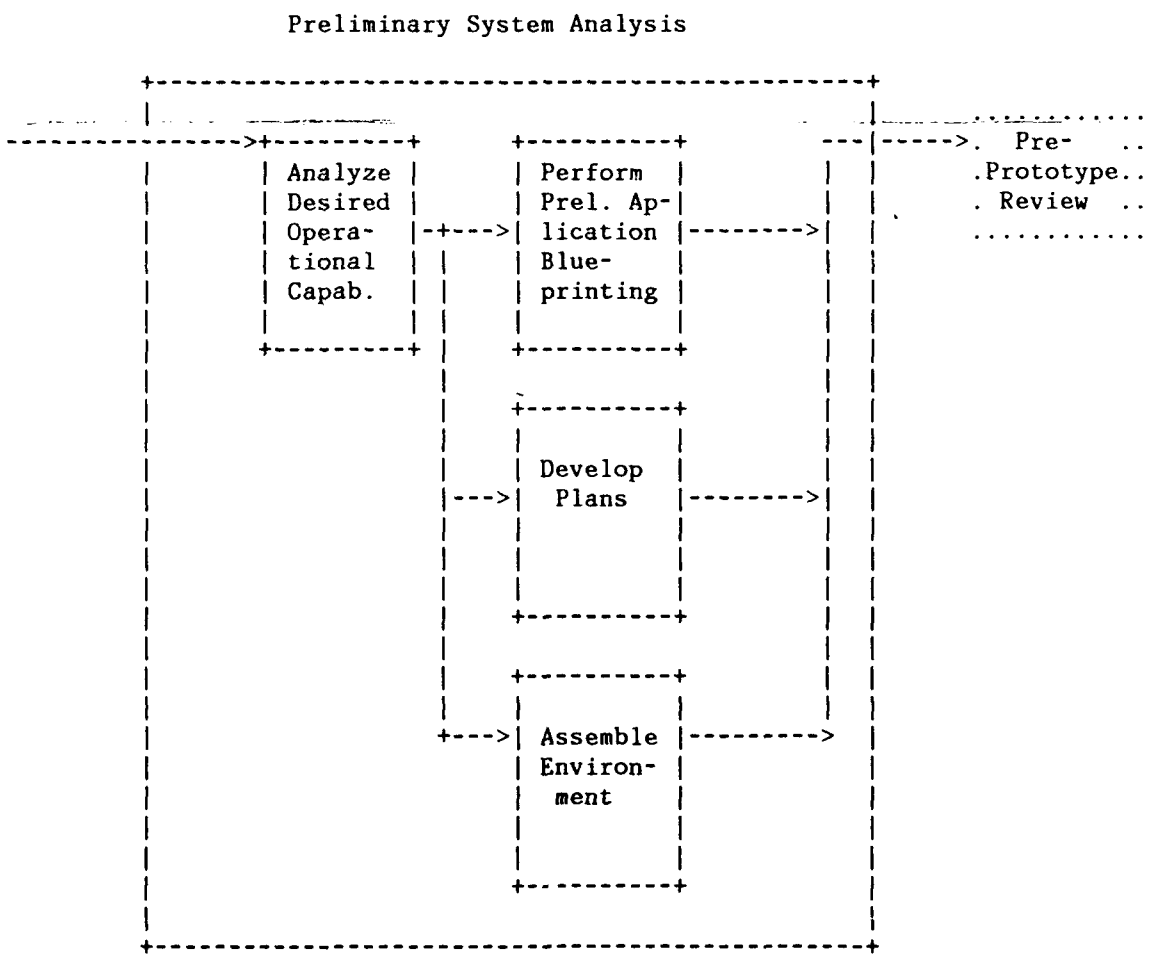


Figure 2. Preliminary System Analysis Tasks

The Analyze Desired Operational Capabilities task formulates a high level understanding of the mission and operational needs of the system. To accomplish this, customer inputs, new technologies and similar systems are analyzed and end users and domain experts are interviewed. The Develop Plans task defines the preliminary strategies for incrementally building the system, for ensuring usability, and for developing system software. It also defines the purpose and evaluation criteria of the inital system prototypes. The Perform Preliminary Application Blueprinting task captures "analysis, design and implementation information for a specific domain in a form which facilitates reuse". (IBM1490) The Assemble Environment task instantiates a base system development environment complete with operating procedures.

The validations performed within the Preliminary System Analysis phase are successful internal reviews of the mission statement, operational need and concept of the system, developed plans and initial prototype capabilities. The internal review of the mission statement and operational need, performed during the Analyze Desired Operational Capabilities task, ensures that a high level view of the system and its capabilities is understood. The internal review of the operational concept, performed during the Perform Application Blueprinting task, ensures that the operational description and preliminary high level architecture satisfy the operational need. The internal review performed during the Develop Plans task ensures the quality and completeness of the plans developed during this phase. The internal review of the initial system prototype capabilities, also performed during the Perform Preliminary Application Blueprinting task, ensures that the system developer has a sufficient understanding of the initial prototype capabilities and how the prototype fits into the evolution of the system.

The exit criteria for the Preliminary System Analysis phase is a successful Pre-Prototyping Review. This review ensures that the system developers have a sufficient understanding of the users' requirements and system operational capabilities to build a prototype. It also ensures an understanding between the customer team and the development team of the life cycle approach, milestones and deliverables to be followed by the project. The review includes the approval of all the outputs shown in the process model figure.

The process models for each of the Preliminary System Analysis tasks are described in the subsections below.

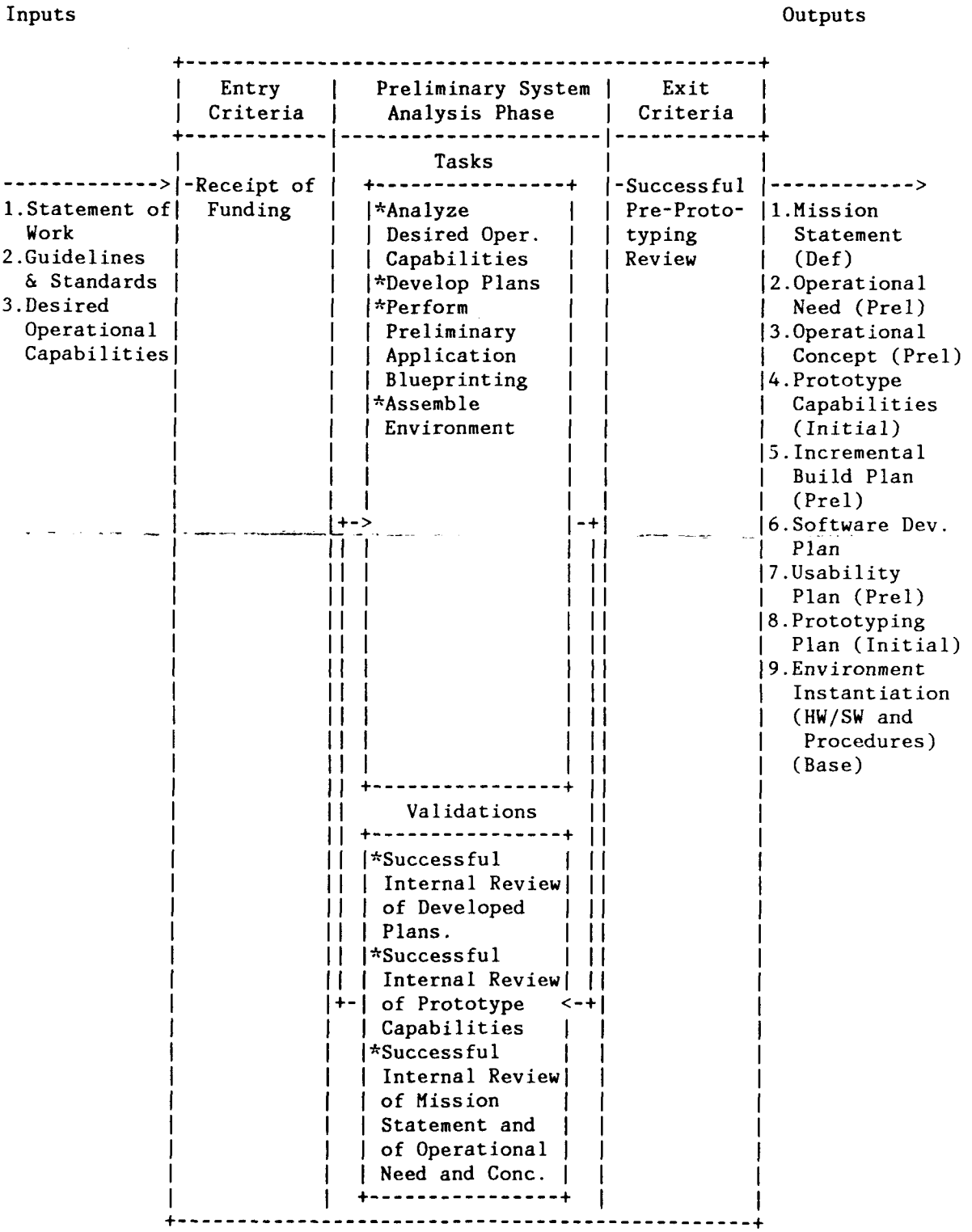


Figure 3. Preliminary System Analysis Process Model

Analyze Desired Operational Capabilities Process Model

The process model for the Analyze Desired Operational Capabilities task is shown in Figure 4 on page 17.

The entry criteria for this task is the receipt of project funding. The inputs, as shown in the figure, are provided by the customer team.

The subtasks performed during the Analyze Desired Operational Capabilities task are:

- Review Current System Information
- Review Reference Material and Similar Systems
- Interview Potential End Users and Domain Experts
- Visit Field Sites
- Define Mission Statement and
- Define Preliminary Operational Need.

The Review Current System Information task surveys and analyzes the existing system (including all supporting documentation) to identify system deficiencies. The Review Reference Material and Similar Systems task surveys alternative methods and systems to identify desireable capabilities. The Interview End Users and Domain Experts task surveys end users for their view of the needed system capabilities and usability features, and consults domain experts to identify capabilities common to all systems within a domain. The Visit Field Sites task enables developers to see the environment in which the system must operate. The Define Mission Statement task relates the mission of the overall system to the desired capabilities of the operational system and defines the purpose and goals of the system. The Record Preliminary Operational Need task uses the results of the other Analyze Desired Operational Capabilities subtasks to develop the statement of the operational problem to be solved, describing it in user operational terminology.

The validations performed within the Analyze Desired Operational Capabilities task are successful internal reviews of the mission statement and operational need. The internal review of the mission statement, performed during the Define Mission Statement subtask, ensures that a high level description of the purpose and goals of the operational system are well understood. Likewise, the internal review of the operational need of the system, performed during the Record Preliminary Operational Need subtask, ensures that the preliminary description of the operational problem to be solved is well understood.

The exit criteria for the Analyze Desired Operational Capabilities task is the completion of the definitized Mission Statement and preliminary Operational Need.

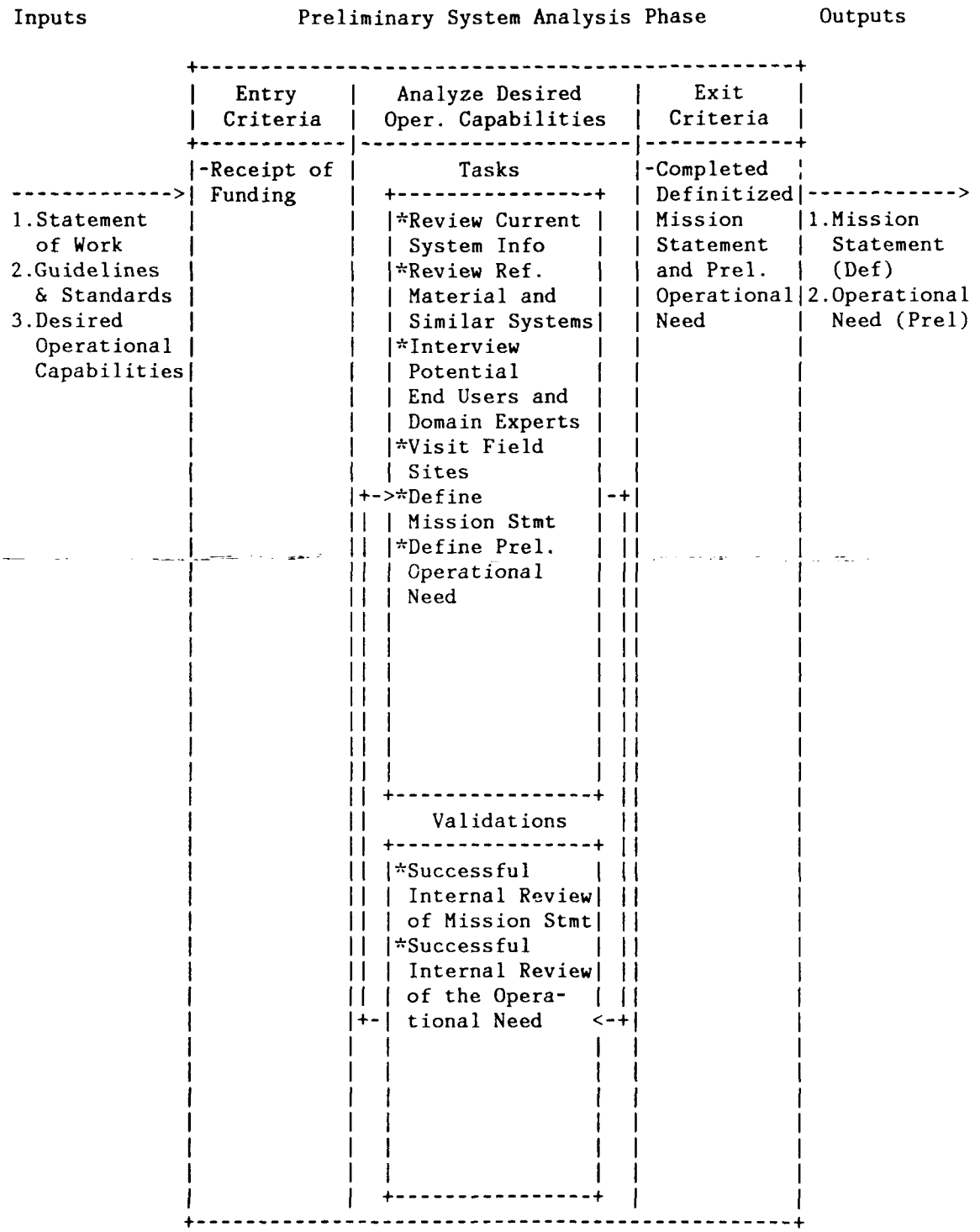


Figure 4. Analyze Desired Operational Capabilities Process Model

Develop Plans Process Model

The process model for the Develop Plans task is shown in Figure 5 on page 19

The entry criteria for this activity is a completed definitized Mission Statement and preliminary Operational Need. The exit criteria of the Analyze Desired Operational Capabilities task satisfies the entrance criteria for this task. The inputs, as shown in the figure, are either received from the customer or are created by the tasks and subtasks of the Preliminary System Analysis Phase.

The subtasks peformed during the Preliminary System Analysis Phase are:

- Develop Preliminary Incremental Build Plan
- Develop Software Development Plan
- Develop Preliminary Usability Plan and
- Develop Prototype Plan for the Initial System Prototype.

The Develop Preliminary Incremental Build Plan subtask defines the initial strategy for evolving system prototypes to an operational system. This includes specifying when and how system capabilities will be incorporated into the system prototypes. The Develop Software Development Plan subtask defines the methods to be used during software development. The Develop Preliminary Usability Plan subtask incorporates the lessons learned from the Analyze Desired Operational Capabilities task and outlines objectives to ensure that the system's user interface meets the needs of the end user. The Develop Prototype Plan subtask defines the approach for building and evaluating the initial system prototypes.

The validations performed within the Develop Plans task are successful internal reviews of the Incremental Build Plan, Software Development Plan, Usability Plan and Prototype Plans developed during this task. The reviews ensure the quality and completeness of each of these plans.

The exit criteria for the Develop Plans task is the completion of the described plans.

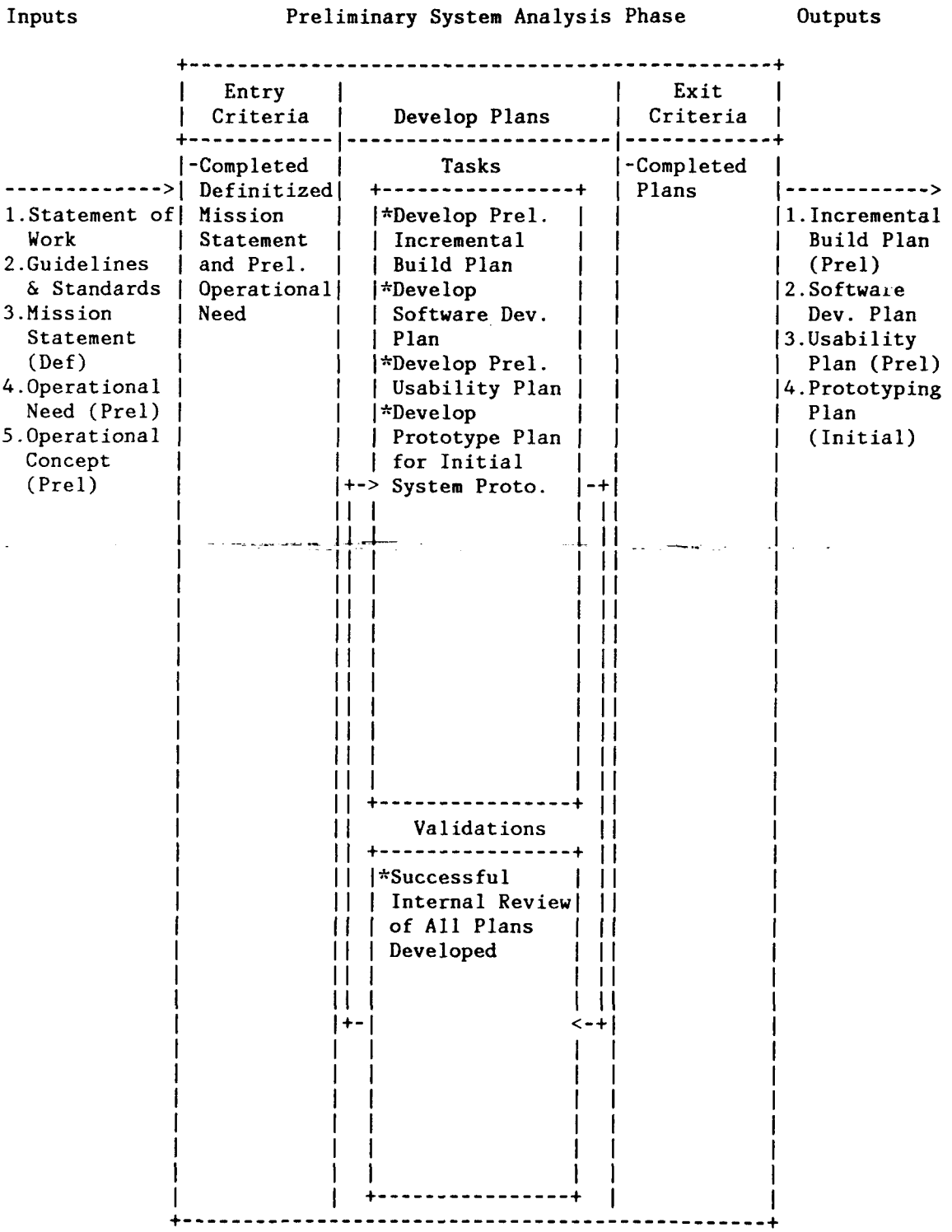


Figure 5. Develop Plans Process Model

Perform Preliminary Application Blueprinting Process Model

The process model for the Perform Preliminary Application Blueprinting task is shown in Figure 6 on page 21

The entry criteria for this activity is a completed definitized Mission Statement and preliminary Operational Need. The exit criteria of the Analyze Desired Operational Capabilities task satisfies the entrance criteria for this task. The inputs, as shown in the figure, are created during the Analyze Desired Operational Capabilities task.

The subtasks performed during the Perform Preliminary Application Blueprinting task are:

- Formalize Domain
- Perform Preliminary Domain Analysis
- Perform Cost/Benefit of Generic Application
- Develop Preliminary Generic/Specific High Level Architecture and
- Develop Initial Prototype Capabilities.

The Formalize Domain subtask establishes the definition and scope of the system domain. The Perform Cost/Benefit of Generic Application subtask justifies whether or not a generic application blueprint should be developed. The Perform Preliminary Domain Analysis subtask identifies the common characteristics and capabilities among systems with the same domain. The Develop Preliminary Generic/Specific High Level Architecture subtask defines the framework for integrating system components. The Develop Initial Prototype Capabilities subtask defines the capabilities to be developed in the initial system prototype(s).

The validations performed within the Perform Preliminary Application Blueprinting task include successful internal reviews of the formalized domain, high level architecture and initial prototype capabilities. The internal review of the formalized domain, performed during the Formalize Domain subtask, ensures that the problem domain is well bounded and understood. The internal review of the high level architecture, performed during the Develop Preliminary Generic/Specific High Level Architecture subtask and the internal review of initial prototype capabilities, performed during the Develop Initial Prototype Capabiltieis subtask, ensure that architecture and initial prototype comply with the objectives outlined in the Incremental Build Plan and Protoyping Plan.

The exit criteria for the Perform Preliminary Application Blueprinting task is the completion of the preliminary application blueprint which includes the creation of the outputs shown in the process model figure.

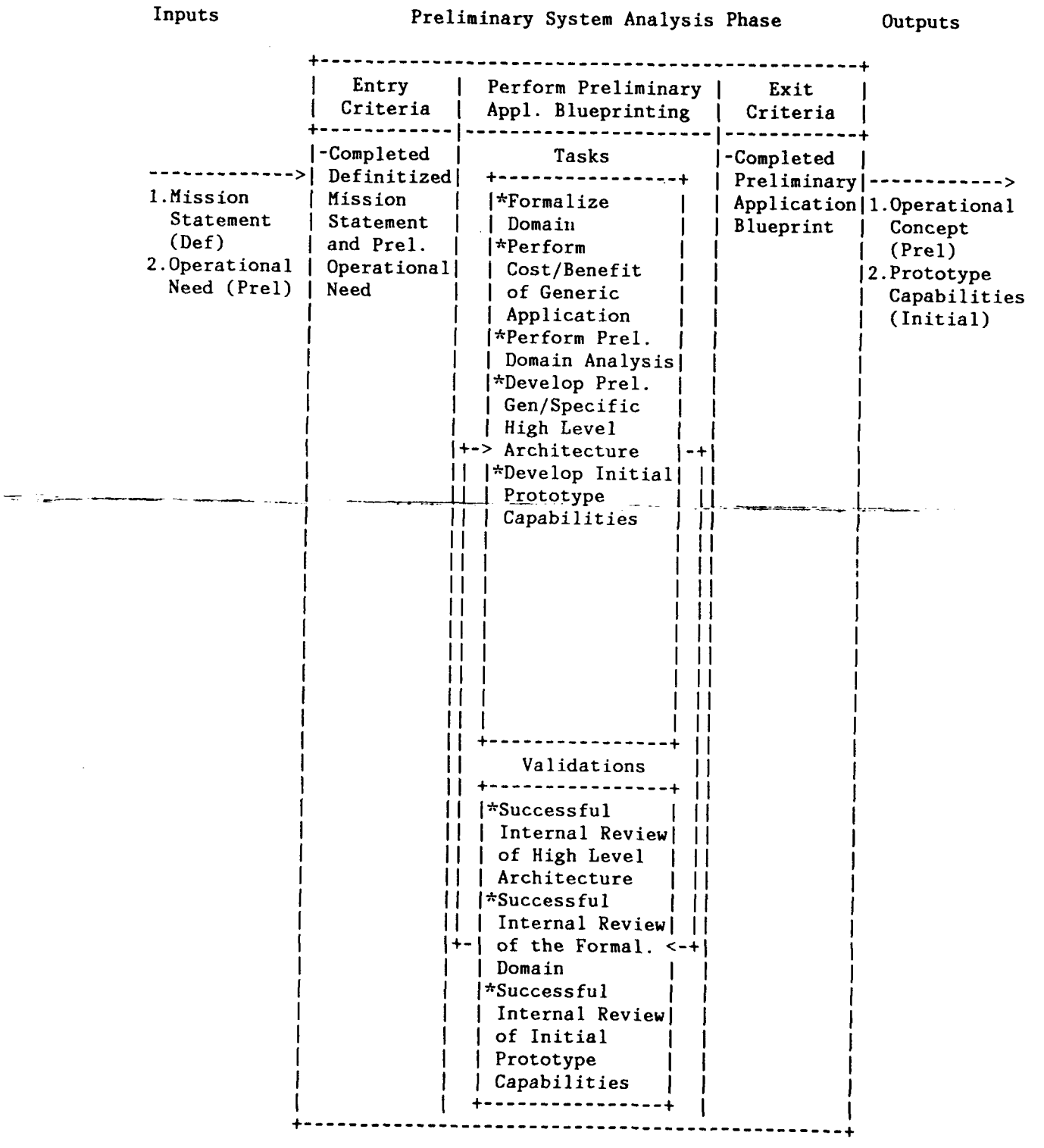


Figure 6. Perform Preliminary Application Blueprinting Process Model

Assemble Environment Process Model

The process model for the Assemble Environment task shown in Figure 7 on page 23

The entry criteria for this activity is a completed definitized Mission Statement and preliminary Operational Need. The exit criteria of the Analyze Desired Operational Capabilities task satisfies the entrance criteria for this task. The inputs, as shown in the figure are either received from the customer or are created by the tasks and subtasks of the Preliminary System Analysis Phase.

The subtasks performed during the Assemble Environment task are:

- Specify Environment Capabilities
- Map Environment Capabilities to Hardware/Software
- Develop Procedures
- Implement Environment and
- Test Environment.

The Specify Environment Capabilities subtask identifies the capabilities that the base environment must have inorder to build the system. The Map Capabilities to Hardware/Software subtask selects the Hardware/Software that will satisfy each of the capabilities needed in the environment. The Develop Procedures subtask defines the procedures for environment operation. The Implement Environment subtask builds the environment, maximizing reuse. This includes using commercial tools where applicable. The Test Environment subtask verifies that the Environment capabilities are satisfied and that they work properly.

The validations performed within the Assemble Environment task include a successful internal review of environment capabilities and procedures and a successful test of the instantiated environment. The internal review of environment capabilities, performed during the Specify Environment Capabilities subtask ensures that the environment will be sufficient for system development. The internal review of the environment procedures, performed during the Develop Procedures subtask, ensures that they are sufficient for operating and maintaining the environmnet. The Test Environment subtask verifies the existence and integrity of the environment capabilities.

The exit criteria for the Assemble Environment Task is an instantiated base environment complete with operating procedures.

Note: As the project progresses, the capabilities of the environment may change as well. The environment should be adapted to the changing needs of the project throughout the life cycle. A concious decision was made not to define a "Maintain Environment" task for each phase because it appeared to detract from the major tasks of each phase.

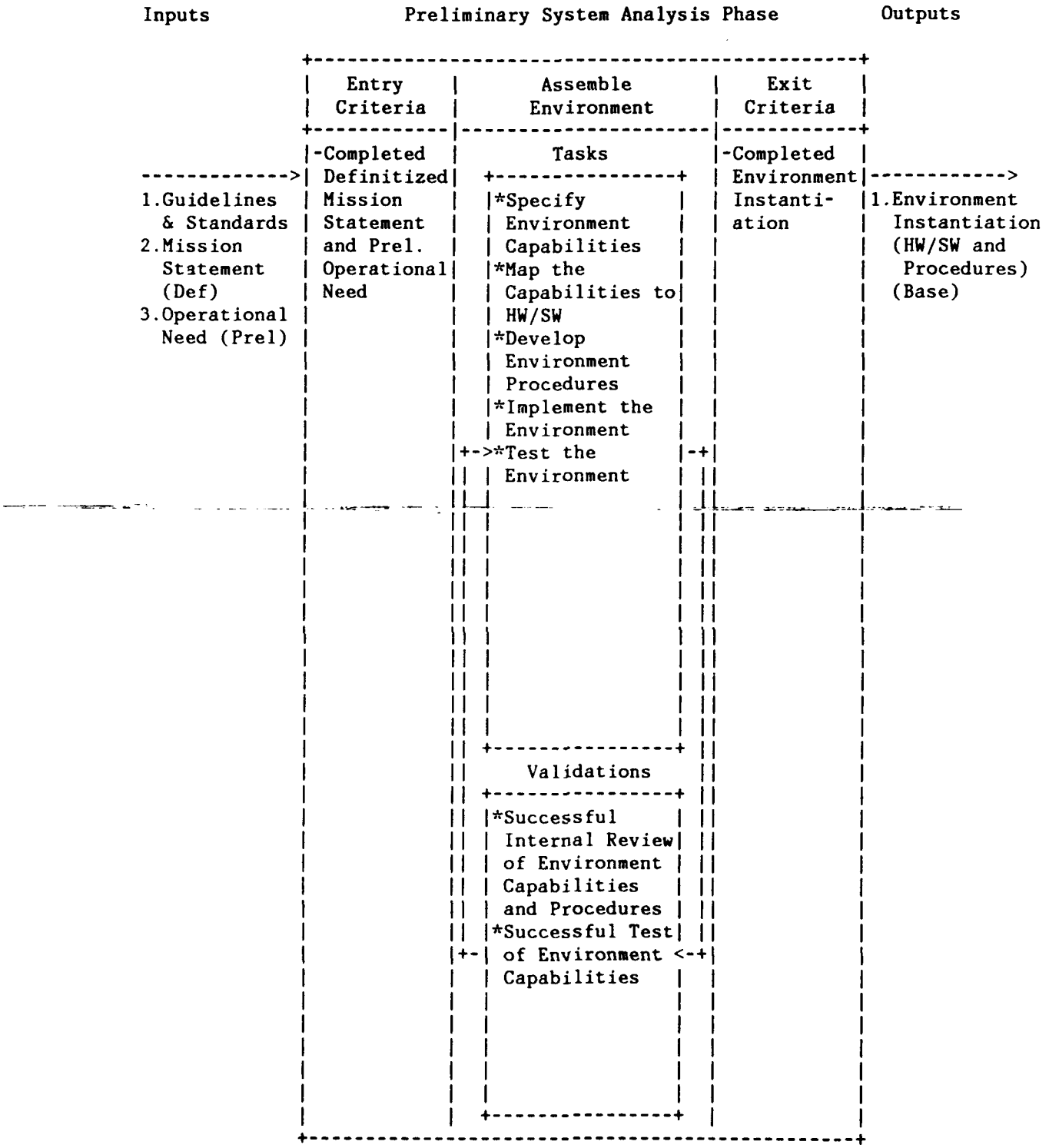


Figure 7. Assemble Environment Process Model

System Architecture Process Model

The process model for the System Architecture phase is shown in Figure 9 on page 26.

The entry criteria for this phase is the successful completion of the Pre-Prototyping Review. This includes approval of all inputs by the customer team. The inputs, as shown in the figure. are created by the Preliminary System Analysis phase.

The tasks performed during the System Architecture phase are:

- Assemble System Prototype
- Evaluate System Prototype
- Perform Expanded System Analysis
- Perform Pre-Productization Analysis.

The relationship between System Architecture tasks is shown in the figure below. For multiple prototypes, many of these tasks would be performed concurrently for each prototype, until a full-capability system prototype is developed.

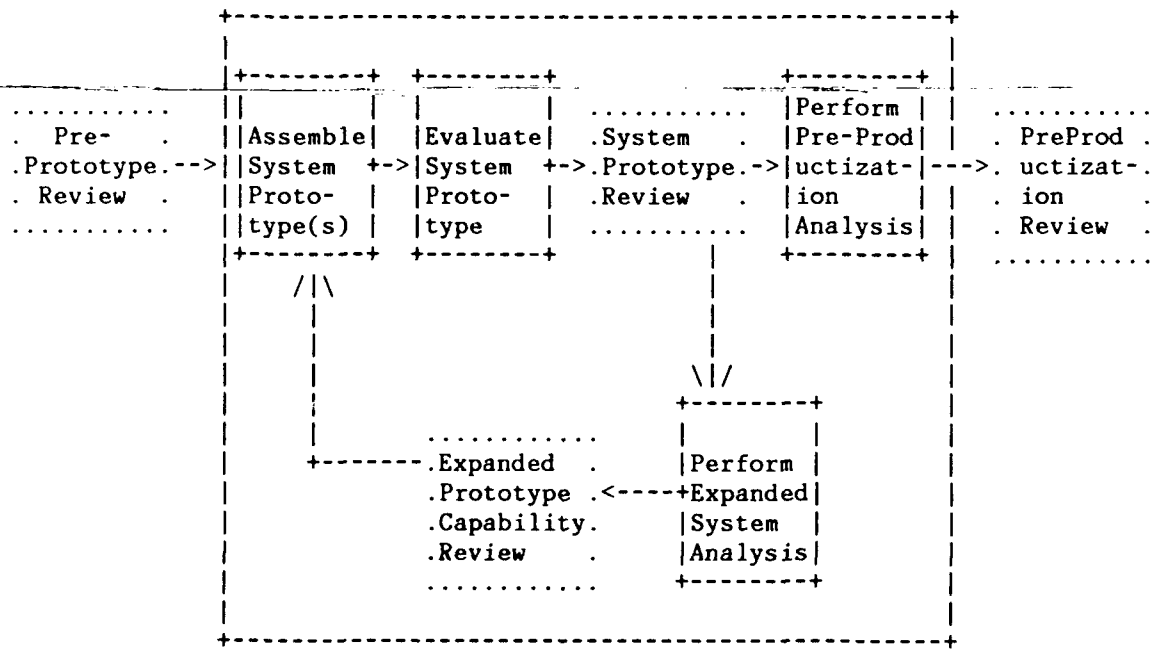


Figure 8. System Architecture

The Assemble System Prototype task builds incremental system prototypes (including protoypes for alternative approaches), primarily from reusable components in the repository. Each incremental prototype is part of the evolution of a full-capability system prototype, which is the last prototype built by this phase. The Evaluate System Prototype task is the task which provides a formal demonstration and evaluation of the incremental system prototype. During this task, planned users of the system perform "hands-on" evaluations of the prototype. The Perform Expanded System Analysis task defines the additional capabilities and planning required for the next increment of the system prototype. The Perform Pre-Productization Analysis task defines the approach to building a productized system from the full-capability system prototype. The definitization of the system definitions is also performed in this task.

Two major validations are performed within the System Architecture phase: System Prototype Review and Expanded Prototype Capability Review. The System Prototype Review provides a validation of the results of the incremental system prototype evaluation. If the review is successful and the incremental system prototype is considered to be a full-capability prototype, then the Perform Pre-Productization Analysis task is entered. If the review is successful and the incremental system prototype is not considered to be a full-capability prototype, then the Perform Expanded System Analysis task is entered. If the review is unsuccessful, then the Assemble System Prototype task is re-entered to correct the prototype deficiencies.

The Expanded Prototype Capability Review provides a validation of the outputs created by the Perform Expanded System Analysis task. If the review is successful, then the Assemble System Prototype task is entered to build the next increment of the system prototype. If the review is unsuccessful, then the Perform Expanded System Analysis task is re-entered to correct the identified deficiencies.

The exit criteria for the System Architecture phase is the conducting of a successful Pre-Productization Review. This includes the approval of all the outputs shown in the process model figure. The outputs include definitized versions of the operational need and operational concept and the required plans to allow the full-capability prototype to be productized. If the review is unsuccessful, then the Perform Pre-Productization Analysis task is re-entered to correct the identified deficiencies.

The process models for each of the System Architecture tasks are described in the subsections below.

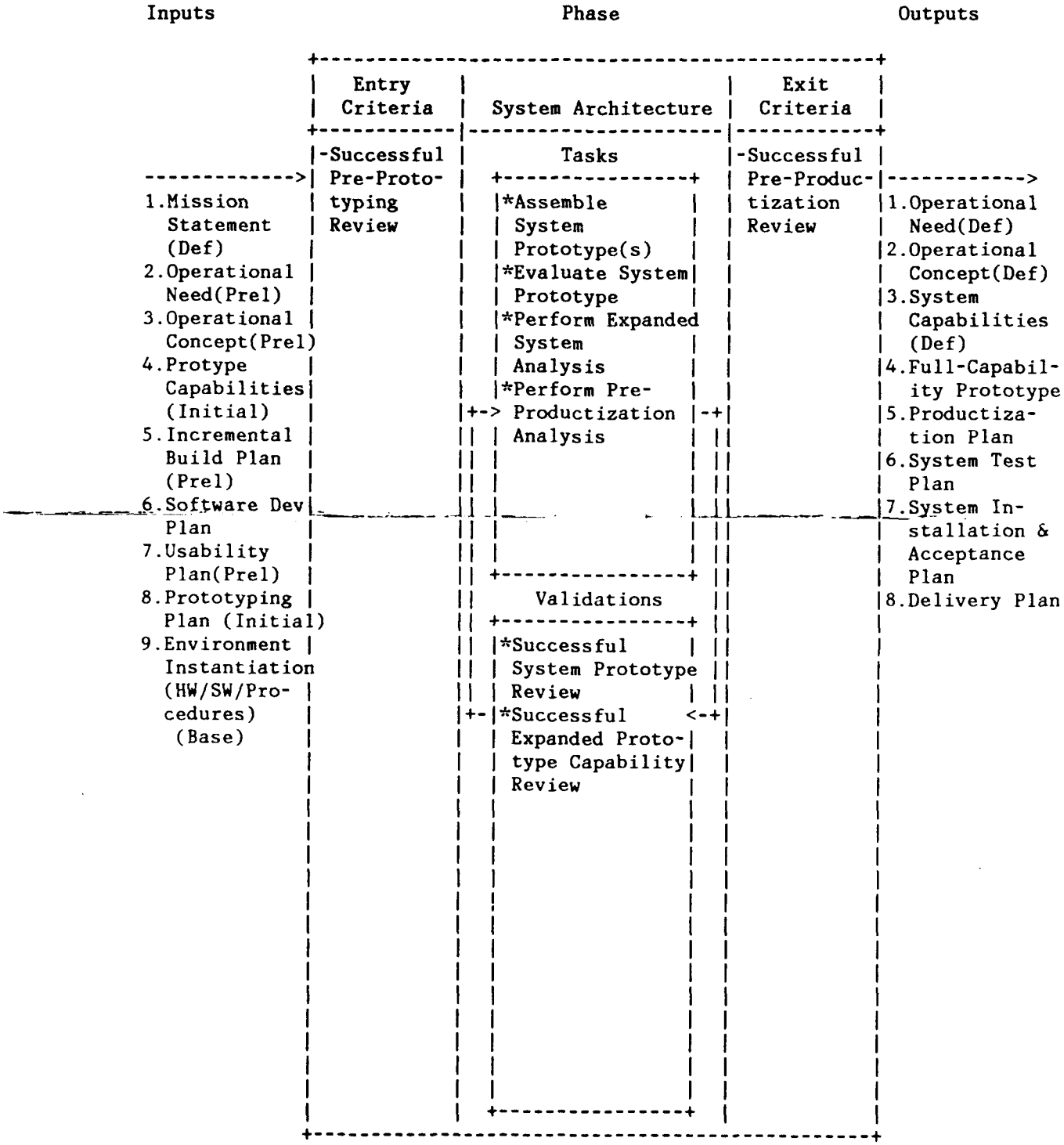


Figure 9. System Architecture Process Model

Assemble System Prototype Process Model

The process model for the Assemble System Prototype task is shown in Figure 10 on page 28.

The entry criteria for this task are either a successful Pre-Prototyping Review or an Expanded Prototype Capability Review. If the entry criteria is a successful Pre-Prototyping Review, then this is the first entry into this task, the inputs are created by the Preliminary System Analysis phase, and they are used to assemble the initial system prototype. If the entry criteria is a successful Expanded Prototype Capability Review, then the inputs to this task are created by the Expanded System Analysis task and are used to assemble a subsequent expanded capabilities prototype or an alternative prototype, e.g., with different capabilities.

Note: Some of the inputs are marked as "(Prel or Exp)", i,e., preliminary or expanded. The preliminary versions of the inputs are created by the Preliminary System Analysis phase. The expanded versions are created by the Expanded System Analysis task.

The subtasks performed during this task are:

- Design System Prototype
- Build System Prototype
- Test System Prototype.

The Design System Prototype subtask performs the system prototype design, with maximum use of existing designs of reusable components. If new components need to be developed, or reusable components need to be adapted, a component development request is generated. If new technology needs to be developed, a new technology request is generated. As required, trade studies are also performed, e.g., hardware/software, size, capacity, performance. The Build System Prototype subtask integrates already existing reusable components, adapted components, and new components into an incremental (initial or expanded) version of the system prototype. The Test System Prototype subtask performs the internal testing of the system prototype in preparation for the Evaluate System Prototype Task.

The validations performed within the Assemble System Prototype task are successful internal reviews of prototype design, prototype test plan, and prototype evaluation plan. The internal review of prototype design is performed within the Design System Prototype subtask to ensure that the design satisfies the prototype capabilities, which are input to this subtask. This review is also intended to ensure that all available reusable components have been identified. The internal reviews of prototype test plan and evaluation plan are performed within the Build System Prototype subtask to ensure the quality and completeness of these plans.

The exit criteria for this task is the completion of the system prototype testing.

Note: Many of the outputs are marked as "(Init or Exp)", i,e., initial or expanded. The initial versions of the outputs are the result of the creation and testing of the initial system prototype. The expanded versions of the outputs are the result of the creation and testing of the expanded system prototype.

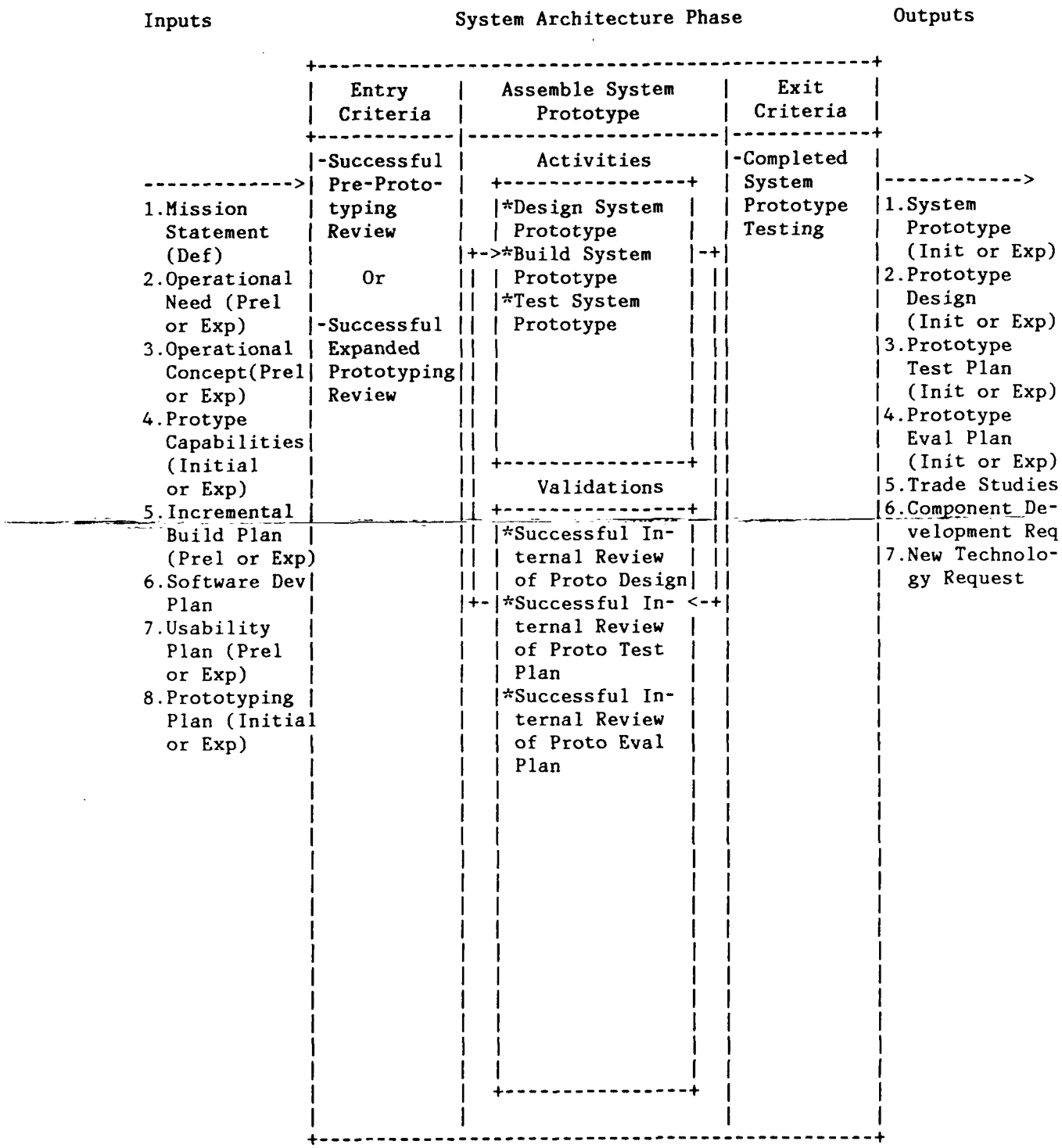


Figure 10. Assemble System Prototype Process Model

Evaluate System Prototype Model

The process model for the Evaluate System Prototype task is shown in Figure 11 on page 30. The entry criteria for this task is completion of testing of the system prototype by the Test System Prototype subtask (of Assemble System Prototype task). The primary inputs are the system prototype and the system prototype evaluation plan. These inputs are either the initial versions for the first time through this task or expanded versions for subsequent times. The other inputs are primarily for reference in support of the perfomance of this task.

The subtasks performed by this task are:

- Develop System Prototype Evaluation Material
- Conduct System Prototype Evaluation
- Create System Prototype Evaluation Results.

The Develop System Prototype Evaluation Material subtask produces material necessary to conduct the evaluation of the system prototype and to allow the users to understand the system prototype well enough evaluate it. This includes evaluation procedures and training and presentation procedures (if needed). The Conduct System Prototype Evaluation subtask performs the actual evaluation of the system prototype. This includes, as required, hands-on evaluation by planned users and monitoring of the users reactions, system responses, and performance during the evaluation. The Create Prototype Evaluation Results subtask produces a detailed description of the results of the evaluation of the system prototype. This includes, as mentioned above, users reactions, system responses, and performance during the evaluation.

The validations performed within this task are internal review of prototype evaluation material, dry run of prototype evaluation, and feedback from all prototype evaluators. The internal review of prototype evaluation material and dry run of prototype evaluation are performed within the Develop System Prototype Evaluation Material subtask to ensure the quality and completeness of the material and the readiness of the evaluation team to conduct the prototype evaluation. The validation that feedback has been received from all prototype evaluators is performed within the Create Prototype Evaluation Results subtask to ensure that the results produced will be complete and reflect the evaluations of all participants.

The exit criteria for this task is the completion of the system prototype evaluation. The major output is the prototype evaluation results in determining the direction of subsequent prototyping. The other outputs are actually used in support of the conduct of the system prototype evaluation.

PROCESS MODEL

System Architecture Phase

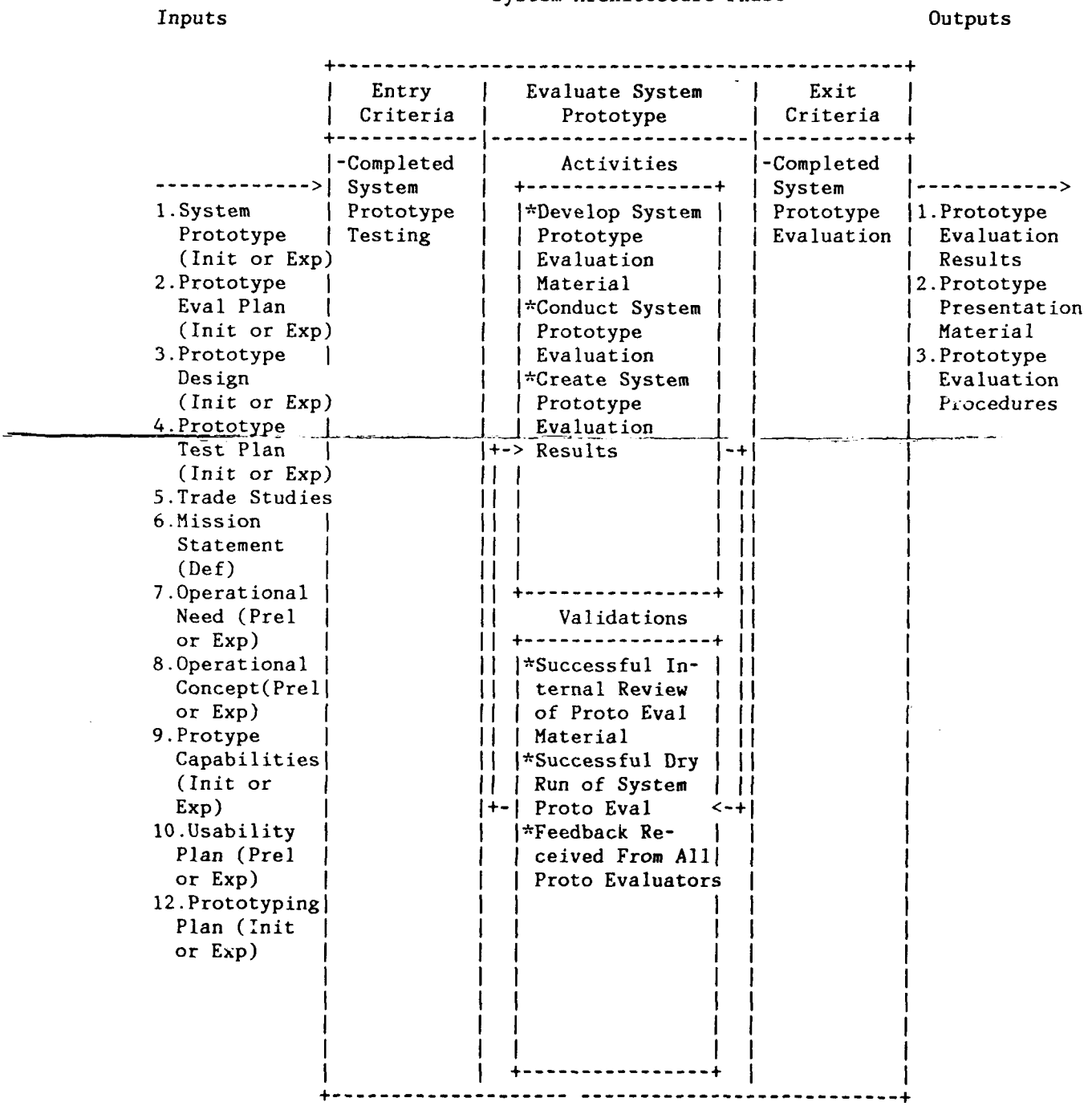


Figure 11. Evaluate System Prototype Process Model

Perform Expanded System Analysis Model

The process model for the Perform Expanded System Analysis task is shown in Figure 12 on page 32. The entry criteria for this task is completion of the System Prototype Evaluation review, during which it was determined that additional prototypes (either expanded or alternatives) need to be developed. The primary input is the system prototype evaluation results which are created by the Evaluate System Prototype task and describe the results of the evaluation of the previous prototype. The other inputs describe either the initial version of the system prototype, for the first time through this task, or expanded versions, for subsequent times.

The subtasks performed by this task are:

- Analyze Prototype Evaluation Results
- Perform Expanded Application Blueprinting
- Develop Expanded Plans.

The Analyze System Prototype Evaluation Results subtask assesses the results of the system prototype evaluation and review, in order to identify what additional capabilities need to be added to the next system prototype to be developed. The Perform Expanded Application Blueprinting subtask expands the preliminary application blueprinting, that was performed as part of the Preliminary System Analysis phase, to include the additional system prototype capabilities. The Develop Expanded Plans subtasks refines the preliminary plans that were produced as part of the Preliminary System Analysis phase, to reflect the development of additional capabilities for the expanded system prototype.

The validations performed within this task are internal reviews of the expanded operational need, operational concept, system prototype capabilities, and plans. The internal reviews of the expanded operational need, operational concept, and system prototype capabilities are performed within the Perform Expanded Application Blueprinting subtask to ensure the quality and completeness of these outputs. The internal review of the expanded plans is performed within the Develop Expanded Plans subtask to ensure that the plans are consistent with one another and are feasible.

The exit criteria for this task is the conducting of a successful Expanded Prototype Capability Review. This includes the approval of all the outputs shown in the process model figure. As mentioned previously, if the review is unsuccessful, then the Perform Expanded System Analysis task is re-entered to correct the identified deficiencies.

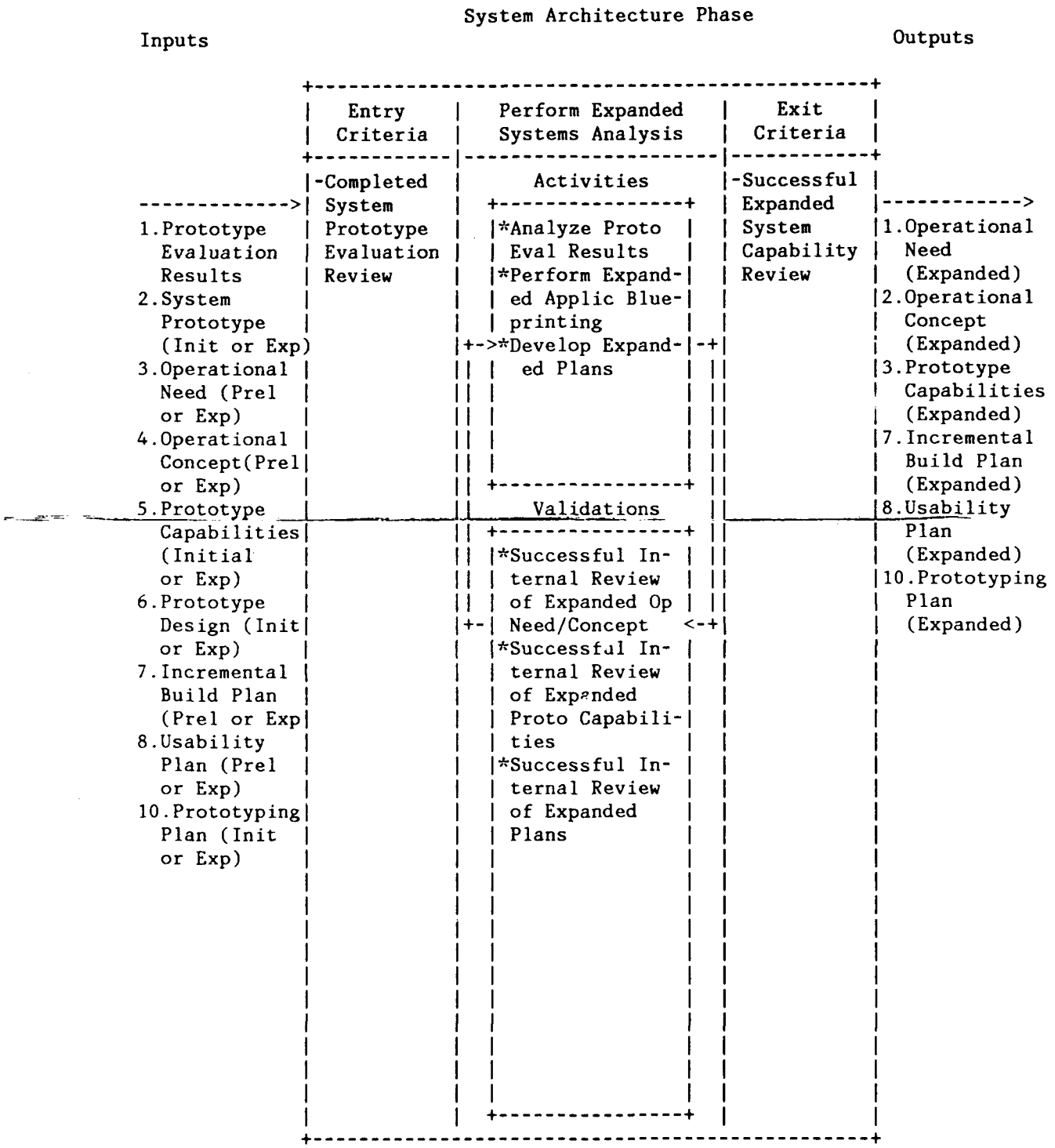


Figure 12. Perform Expanded System Analysis Process Model

Perform Pre-Productization Analysis Model

The process model for the Perform Pre-Productization Analysis task is shown in Figure 13 on page 34. The entry criteria for this task is completion of the System Prototype Evaluation review, during which it is determined that the current system prototype is a full-capability prototype. The primary inputs are the system prototype evaluation results and the expanded descriptions and plans for the system and the full-capability prototype.

The subtasks performed by this task are:

- Analyze Prototype Evaluation Results
- Develop Definitized System Definitions
- Develop Productization Approach
- Develop Definitized Plans.

The Analyze System Prototype Evaluation Results subtask assesses the results of the system prototype evaluation and review, in order to identify what additional capabilities need to be added to the productized system to be developed. The Develop Definitized System Definitions subtask creates definitized versions of the operational need, operational concept, and system capabilities. The Develop Productization Approach subtask defines the strategies for performing the productization of the full-capability prototype. The Develop Definitized Plans subtask produces plans to support productization, test, acceptance, and delivery of the system.

The validations performed within this task are internal reviews of the definitized operational need, operational concept, system capabilities, and plans. The internal reviews of the definitized operational need, operational concept, and system capabilities are performed to ensure the quality and completeness of the productized system as reflected in these products. The internal review of the definitized plans is performed to ensure that the plans are consistent with one another and are feasible.

The exit criteria for this task is the conducting of a successful Pre-Productization Review. This includes the approval of all the outputs shown in the process model figure. The outputs include definitized versions of the operational need and operational concept and the required plans to allow the full-capability prototype to be productized. As mentioned previously, if the review is unsuccessful, then the Perform Pre-Productization Analysis task is re-entered to correct the identified deficiencies.

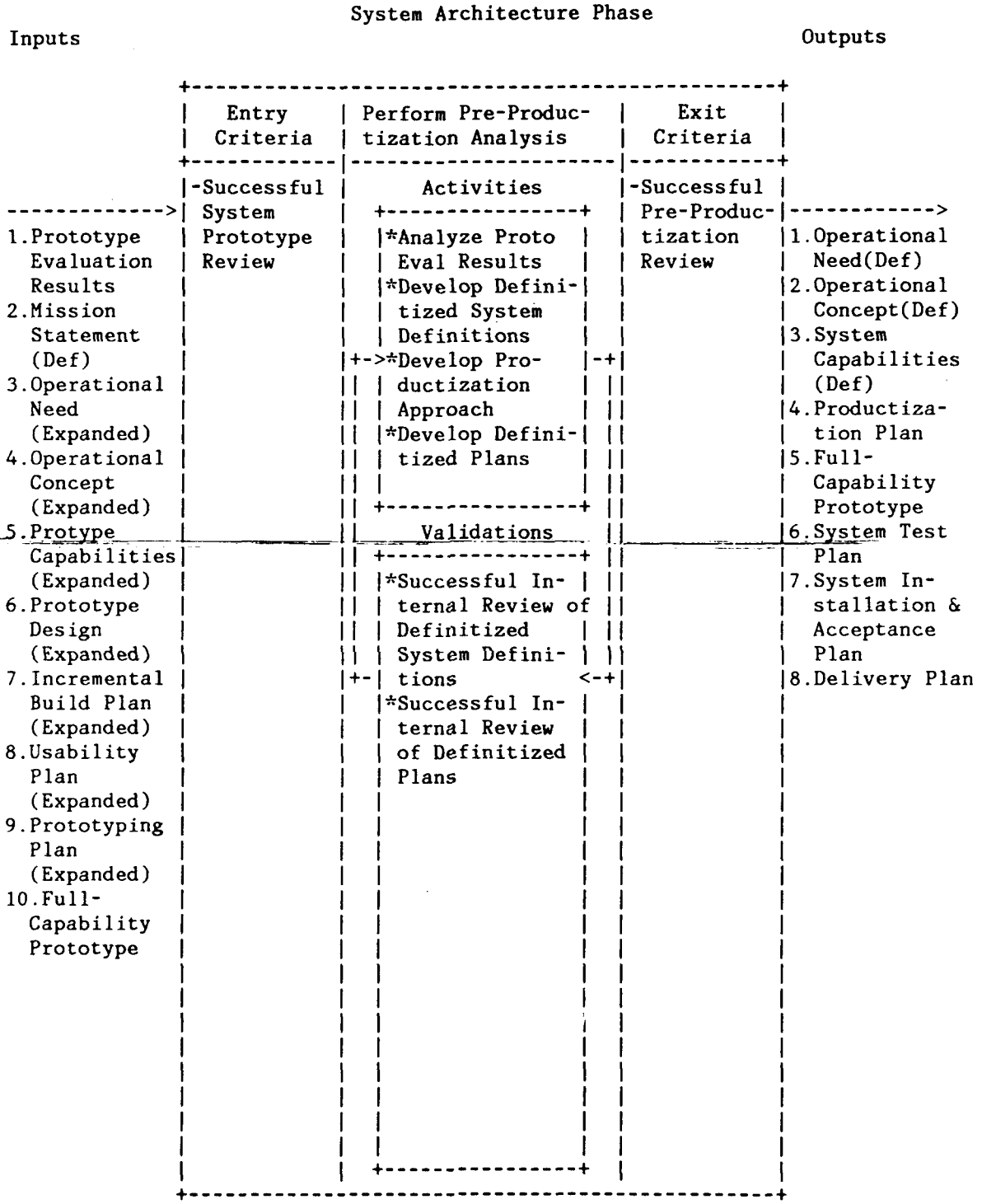


Figure 13. Perform Pre-Productization Analysis Process Model

Software Growing Process Model

The process model for the Software Growing Process phase is shown in Figure 15 on page 37.

The entry criteria for this phase is the receipt of either a Component Development Request or New Technology Request. These inputs are the primary inputs to the Software Growing phase. They are either from the Preliminary System Analysis phase, for early (in the life cycle) requests for component and new technology development, or from the System Architecture phase or Develop New Technology task (within Software Growing) for later requests. Component Development Request can also be received from the Productization and Production phase. The other inputs are primarily for reference in support of the of the associated request.

Note: Some of the inputs are marked as "(Prel or Exp)" or "(Init or Exp)". The Prel (preliminary) or Init (initial) versions of the inputs are created by the Preliminary System Analysis phase. The Exp (expanded) versions are created by the System Architecture phase.

The tasks performed during the Software Growing phase are:

- Develop Component Prototype(s)
- Evaluate Component Prototype
- Productize Component Prototype
- Perform Component Test and Acceptance
- Develop New Technology.

The relationship between Software Growing tasks is shown in the figure below. For multiple components, many of these tasks would be performed concurrently for each component prototype, until productized (reusable) components are developed.

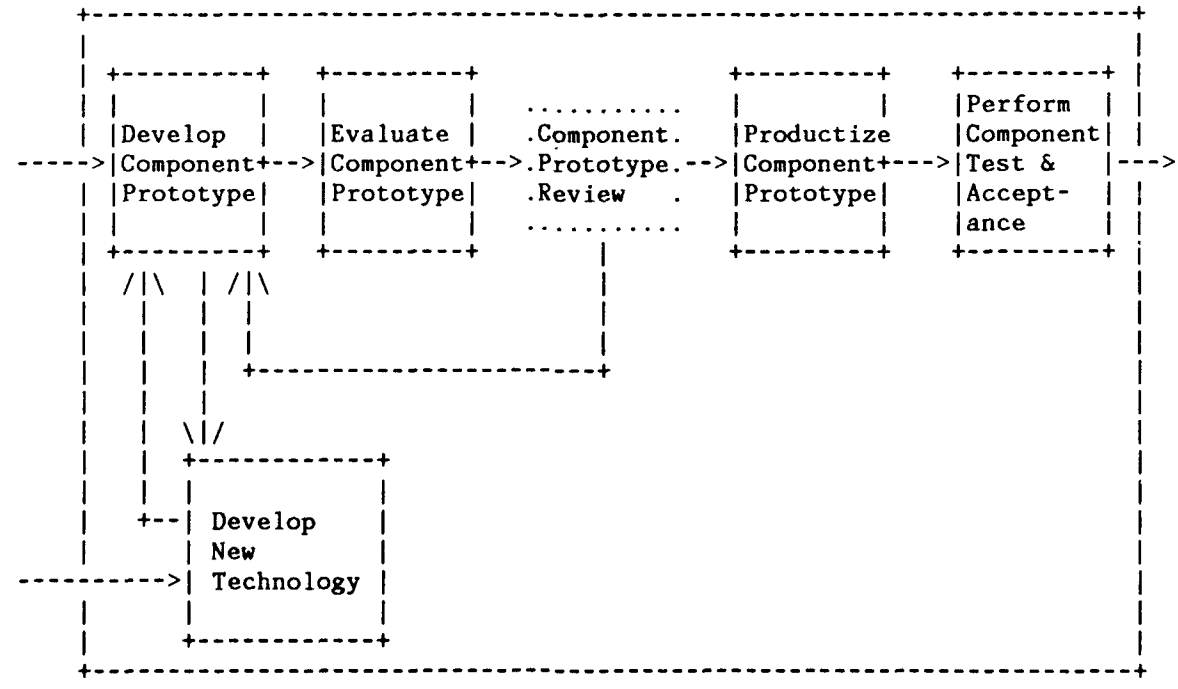


Figure 14. Software Growing Process Model

The Develop Component Prototype task, upon receipt of Component Development Request(s), builds incremental component prototypes (including protoypes for alternative approaches), with maximum use of reusable components in the repository. Each incremental prototype is part of the evolution of a full-capability component prototype. The Evaluate Component Prototype task provides a formal demonstration and evaluation of the incremental component prototype. During this task, planned users of the component perform "hands-on" evaluations of the prototype. The Perform Expanded Component Analysis task defines the additional capabilities and planning required for the next increment of the component prototype. The Productize Component Prototype task converts the full-capability component prototype into a productized component. The definitization of the component capabilities and design is also performed in this task. The Perform Component Test and Acceptance task accomplishes the required verifications of the component prior to entry as a reusable component into the repository.

Note: To allow early use of components to assemble a system prototype within the System Architecture phase, intermediate versions of component prototypes, developed in the Software Growing phase, can be added to the repository.

The Develop New Technology task, upon receipt of the New Technology Request, performs the required analysis, research, and experimentation to develop the requested technology. When a technology is developed to the point where the associated component can be built, a Component Development Request is generated.

The major validation performed within the Software Growing phase is the Component Prototype Review. This review provides a validation of the results of the incremental component prototype evaluation. If the review is successful and the incremental Component prototype is considered to be a full-capability prototype, then the Productize Component Prototype task is entered. If the review is successful and the incremental Component prototype is not considered to be a full-capability prototype, then the Develop Component Prototype task is re-entered to build the next increment of the component prototype. If the review is unsuccessful, then the Develop Component Prototype task is re-entered to correct the prototype deficiencies.

The exit criteria for the Software Growing phase is the conducting of a successful component test and acceptance. This includes the approval of all the outputs shown in the process model figure. The outputs include definitized versions of the component capabilities and design. If the component test and acceptance is unsuccessful, the Productize Component Prototype task is re-entered to correct the identified deficiencies.

The process models for each of the Software Growing tasks are described in the subsections below.

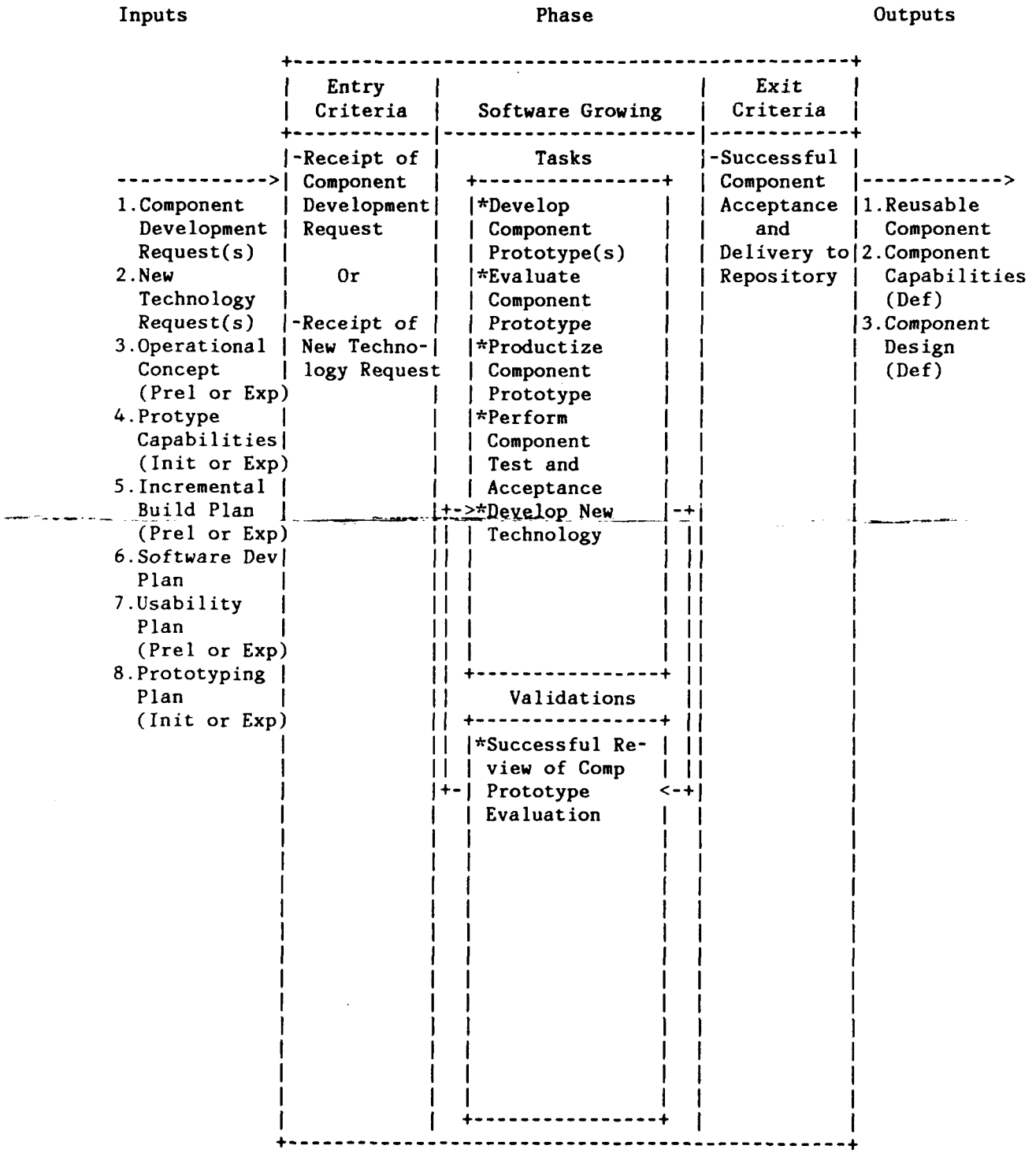


Figure 15. Software Growing Process Model

Develop Component Prototype Process Model

The process model for the Develop Component Prototype task is shown in Figure 16 on page 39.

The entry criteria for this task is either receipt of Component Development Request(s) or receipt of component prototype evaluation results. If the entry criteria is receipt of a Component Development Request, then this is the first entry into this task to develop the requested component and an initial component prototype needs to be built. If the entry criteria is receipt of component prototype evaluation results, then a Component Prototype Review has determined that either an expanded capabilities component prototype needs to be built or deficiencies in the previous component prototype need to be corrected. The other inputs (i.e., in addition to Component Development Requests and component prototype evaluation results) are supporting information, created in other phases and used primarily for reference in support of component prototype development.

The subtasks performed during this task are:

- Design Component Prototype
- Develop Component Plans
- Build Component Prototype
- Test Component Prototype.

The Design Component Prototype subtask specifies the component prototype capabilities and performs the component prototype design, with maximum use of existing reusable components. As required, trade studies are performed, e.g., size, capacity, performance. If new technology needs to be developed, a New Technology Request is generated. The Develop Component Plans subtask creates the component build plan, along with the component prototype test and evaluation plans. The Build Component Prototype subtask creates an incremental (initial or expanded) version of the component prototype. This is accomplished by developing new components, adapting already existing reusable components, and/or using already existing lower level reusable components (if available). The Test Component Prototype subtask performs the internal testing of the component prototype in preparation for the Evaluate Component Prototype Task.

The validations performed within the Develop Component Prototype task are successful internal reviews of component prototype capabilities and design, build plan, test plan, and evaluation plan. The internal review of component prototype capabilities and design is performed within the Design Component Prototype subtask to ensure that the design satisfies the prototype capabilities. This review is also intended to ensure that all available reusable lower level components have been identified. The internal reviews of component prototype test and evaluation plans are performed within the Develop Component Plans subtask to ensure the quality and completeness of these plans.

The exit criteria for this task is the completion of the component prototype testing.

Note: Many of the outputs are marked as "(Init or Exp)", i,e., initial or expanded. The initial versions of the outputs are the result of the creation and testing of the initial component prototype. The expanded versions of the outputs are the result of the creation and testing of the expanded component prototype.

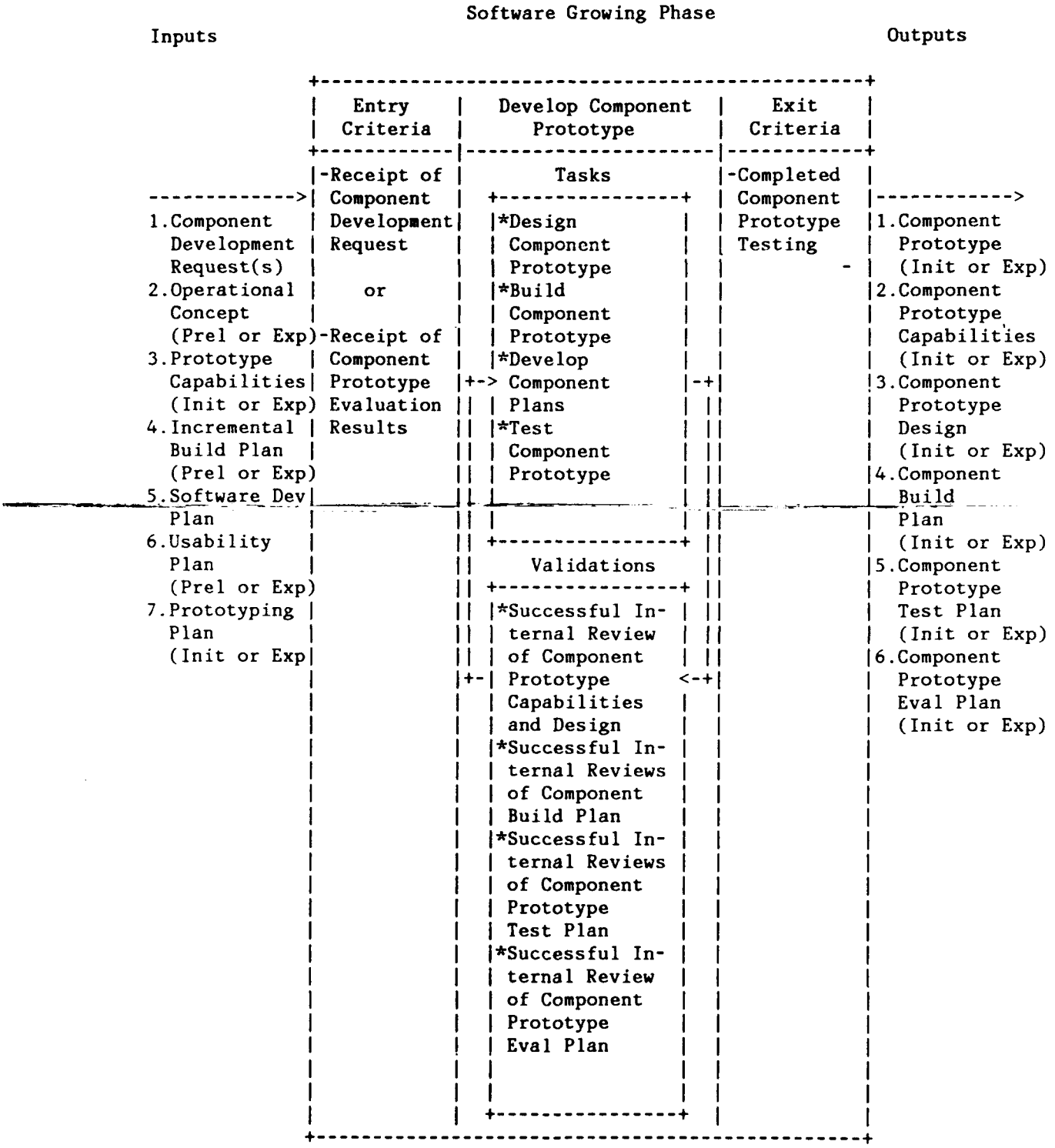


Figure 16. Develop Component Prototype Process Model

Evaluate Component Prototype Model

The process model for the Evaluate Component Prototype task is shown in Figure 17 on page 41.

The entry criteria for this task is completion of testing of the component prototype by the Test Component Prototype subtask (of Develop Component Prototype task). The primary inputs are the component prototype and the component prototype evaluation plan. These inputs are either the initial versions for the first time through this task or expanded versions for subsequent times. The other inputs are primarily for reference in support of the perfomance of this task.

The subtasks performed during this task are:

- Develop Component Prototype Evaluation Material
- Conduct Component Prototype Evaluation
- Create Component Prototype Evaluation Results.

The Develop Component Prototype Evaluation Material subtask produces material necessary to conduct the evaluation of the component prototype and allow the participants to understand the Component prototype well enough to evaluate it. This includes evaluation procedures and presentation and training material (if needed). The Conduct Component Prototype Evaluation subtask performs the actual evaluation of the component prototype. This includes, as required, hands-on evaluation by planned users and monitoring of the users reactions, component responses, and performance during the evaluation. The Create Prototype Evaluation Results subtask produces a detailed description of the results of the evaluation of the component prototype.

The validations performed within this task are internal review of prototype evaluation material, dry run of prototype evaluation, and feedback from all prototype evaluators. The internal review of prototype evaluation material and dry run of prototype evaluation are performed within the Develop Component Prototype Evaluation Material subtask to ensure the quality and completeness of the material and the readiness of the evaluation team to conduct the prototype evaluation. The validation that feedback has been received from all prototype evaluators is performed within the Create Component Prototype Evaluation Results subtask to ensure that the results produced will be complete and reflect the evaluations of all participants.

The exit criteria for this task is the completion of the component prototype evaluation. The major output is the prototype evaluation results, which is used by the Develop Component Prototype subtask to determine the direction of subsequent prototyping. The other outputs are actually used in support of the conduct of the component prototype evaluation.

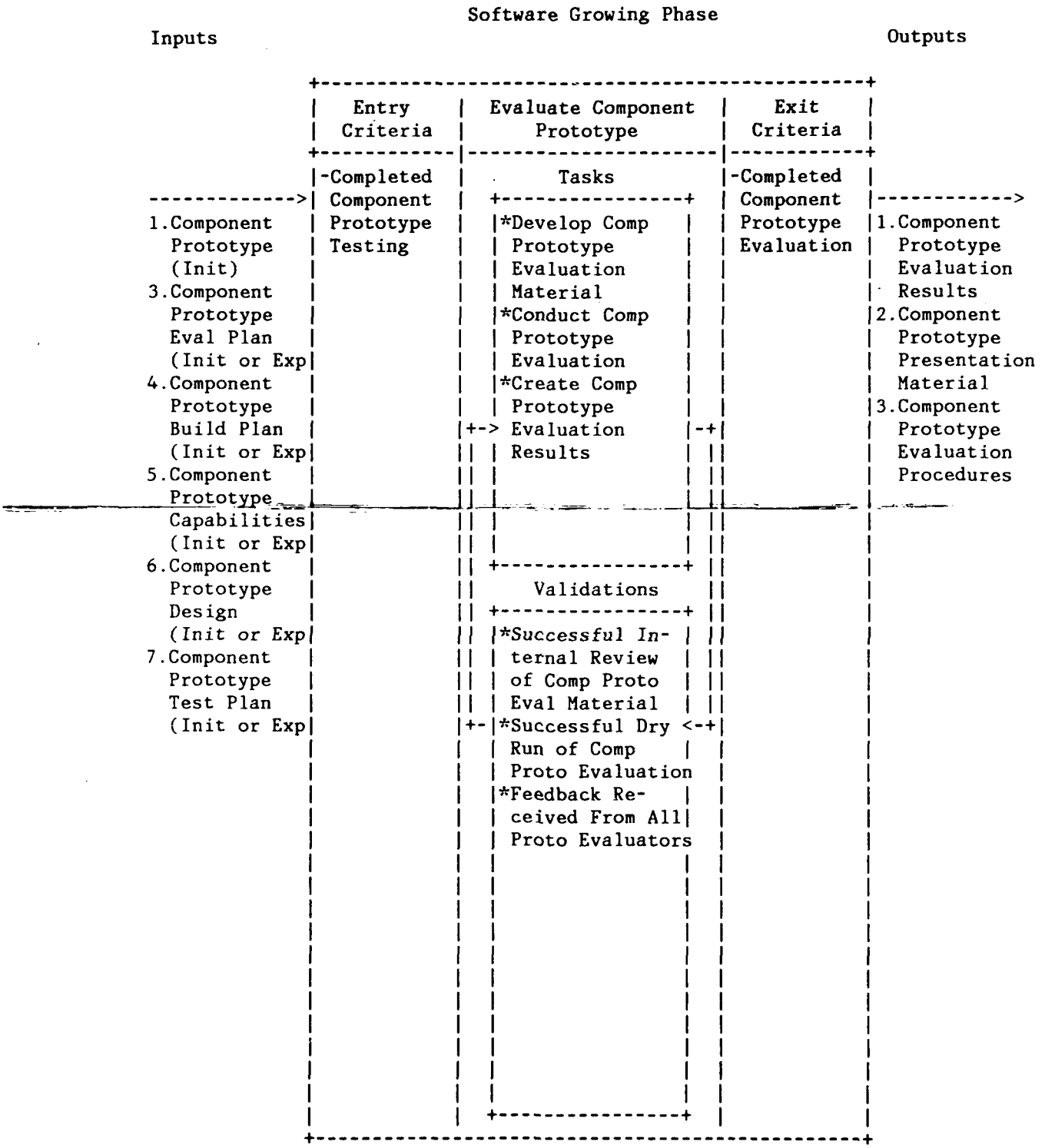


Figure 17. Evaluate Component Prototype Process Model

Productize Component Prototype

The process model for the Productize Component Prototype task is shown in Figure 18 on page 43.

The entry criteria for this task is completion of the Component Prototype Evaluation review, during which it was determined that the current Component prototype is a full-capability prototype. The primary inputs are the component prototype evaluation results, the expanded descriptions and plans for the Component, and the full-capability prototype.

The subtasks performed during this task are:

- Analyze Component Prototype Evaluation Results
- Develop Definitized Component Definitions
- Build Productized Component
- Develop Component Test and Acceptance Plans.

The Analyze Component Prototype Evaluation Results subtask assesses the results of the component prototype evaluation and review, in order to identify what additional capabilities need to be added to the productized component to be developed. The Develop Definitized Component Definitions subtask creates definitized versions of the component capabilities and design. The Build Productized Component subtask converts the full-capability component prototype into a productized, reusable component. The Develop Component Test and Acceptance Plan produces plans to support test and acceptance of the productized component.

The validations performed within this task are internal reviews of the definitized capabilities and design and component test and acceptance plan, and internal testing of the the productized component. The internal reviews of the definitized component capabilities and design are performed to ensure the quality and completeness of the productized component as reflected in these products. The internal review of the component test and acceptance plan is performed to ensure that the plan is complete and is consistent with established standards. The internal testing of the the productized component is performed in preparation for formal component test and acceptance.

The exit criteria for this task is the completion of component productization. This includes the creation of all the outputs shown in the process model figure. The outputs include productized component, definitized versions of component capabilities and design, and component test and acceptance plans.

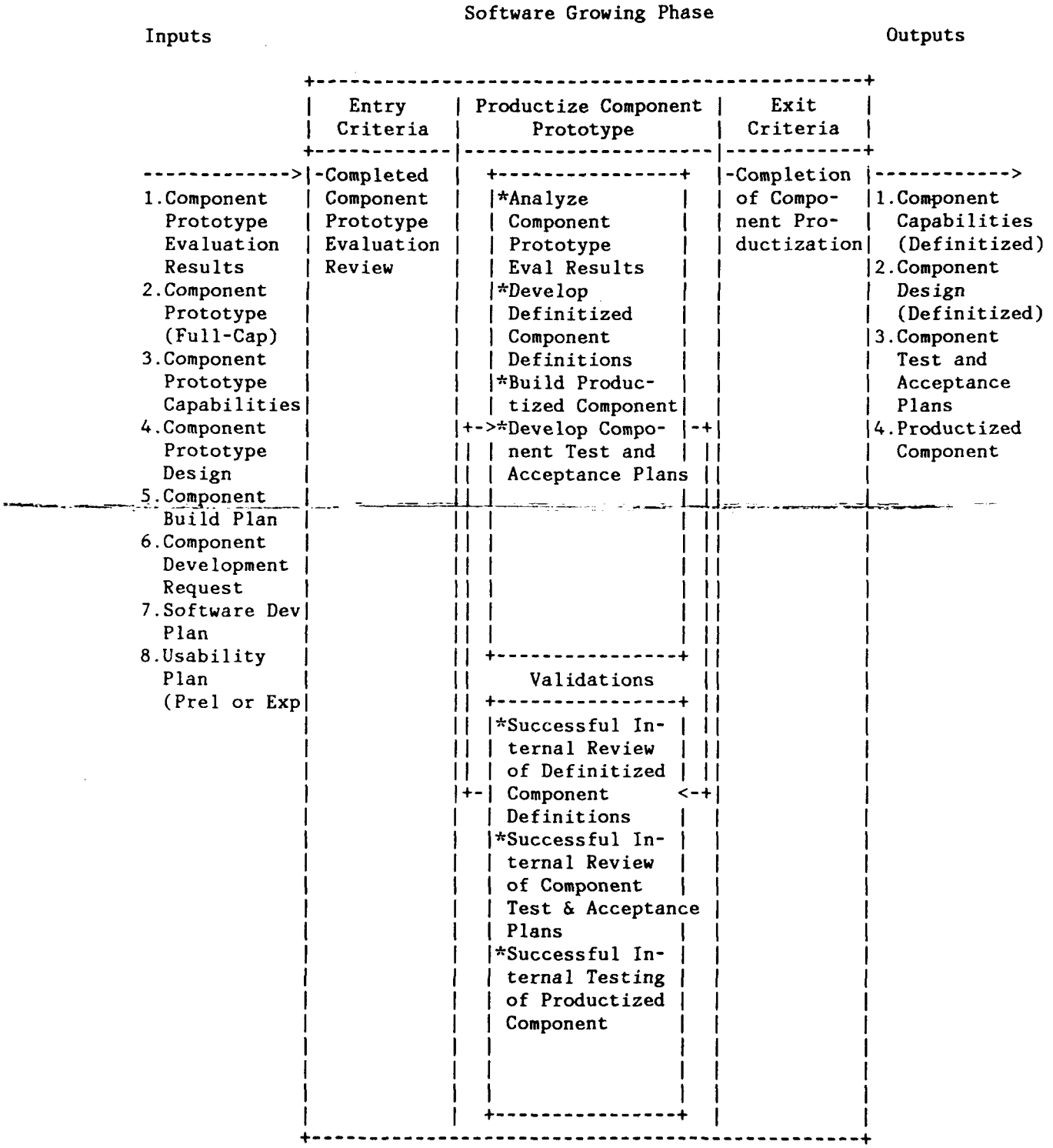


Figure 18. Productize Component Prototype Process Model

Perform Component Test and Acceptance

The process model for the Perform Component Test and Acceptance task is shown in Figure 19 on page 45.

The entry criteria for this task is completion of the component productization, i.e., a productized component is ready to be tested. The inputs include the productized component, and its associated component capabilities and design and component test and acceptance plan.

The subtasks performed during this task are:

- Perform Component Testing
- Perform Component Acceptance
- Deliver Prototype to Repository.

The Perform Component Testing subtask verifies the existence and integrity of component capabilities. The Perform Component Acceptance subtask determines, via component filtering, whether the component meets the established standards for reusability. The Deliver Component subtask ensures that the productized component and all supporting material have been delivered for storage in the repository.

~~The validation performed within this task is successful completion of component testing.~~

The exit criteria for this task is successful completion of component acceptance and deliverye of the productized component to the repository. This makes it available to the System Architecture phase for integration into a system prototype. The outputs include the productized component, definitized versions of component capabilities and design, and component test and acceptance plans.

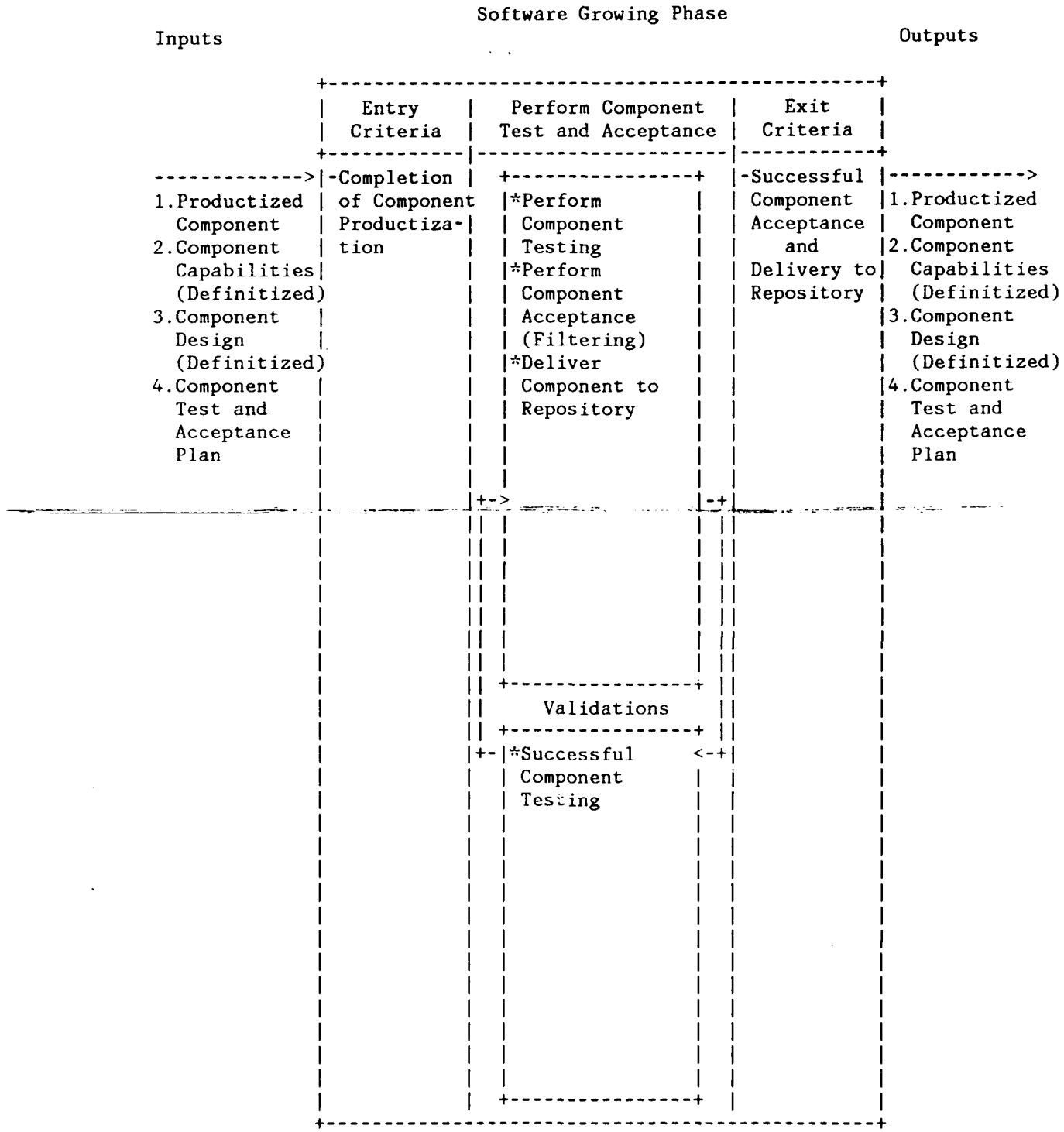


Figure 19. Perform Component Test and Acceptance Process Model

Develop New Technology

The process model for the Develop New Technology task is shown in Figure 20 on page 47. The entry criteria for this phase is the receipt of New Technology Request(s). These requests are either from the Preliminary System Analysis phase, for early (in the life cycle) requests for new technology, or from the System Architecture phase for later requests. The other inputs are primarily for reference in support of development of the new technology.

The subtasks performed during this task are:

- Analyze Technology Base
- Perform Concept Definitions
- Perform Trade Studies
- Perform Pre-Prototype Experimentation
- Develop Conceptual Prototpes
- Perform Basic Research
- Perform Breakthrough Initiatives

The Analyze Technology Base subtask evaluates the current state-of-art to determine to what extent existing technology can meet the request for new technology development. The Perform Concept Definitions subtask defines the concepts, including alternative ones, needed to develop the new technology. The Perform Trade Studies subtask performs technology trade studies to determine what directions the new technology will take. The Perform Pre-Prototype Experimentation subtask provides early experimentation with the new technologies prior to building conceptual prototypes. The Develop Conceptual Prototypes subtasks builds conceptual prototypes to allow hands-on evaluation of the new technology being developed. The Perform Basic Research subtask provides long term research as required, to allow more fundamental underlying technologies to be developed. The Perform Breakthrough Initiatives subtask provides long term development of innovative ideas that have the potential for providing significant improvements to the system being developed, e.g., performance or cost.

The validations performed within this task are not defined at this time.

The exit criteria for this task is the development of the requested technology, to a level of maturity where a Component Development Request can be generated to allow the corresponding component to be built. The outputs, in addition to the Component Development Request, includes the various findings associated with the new technology development.

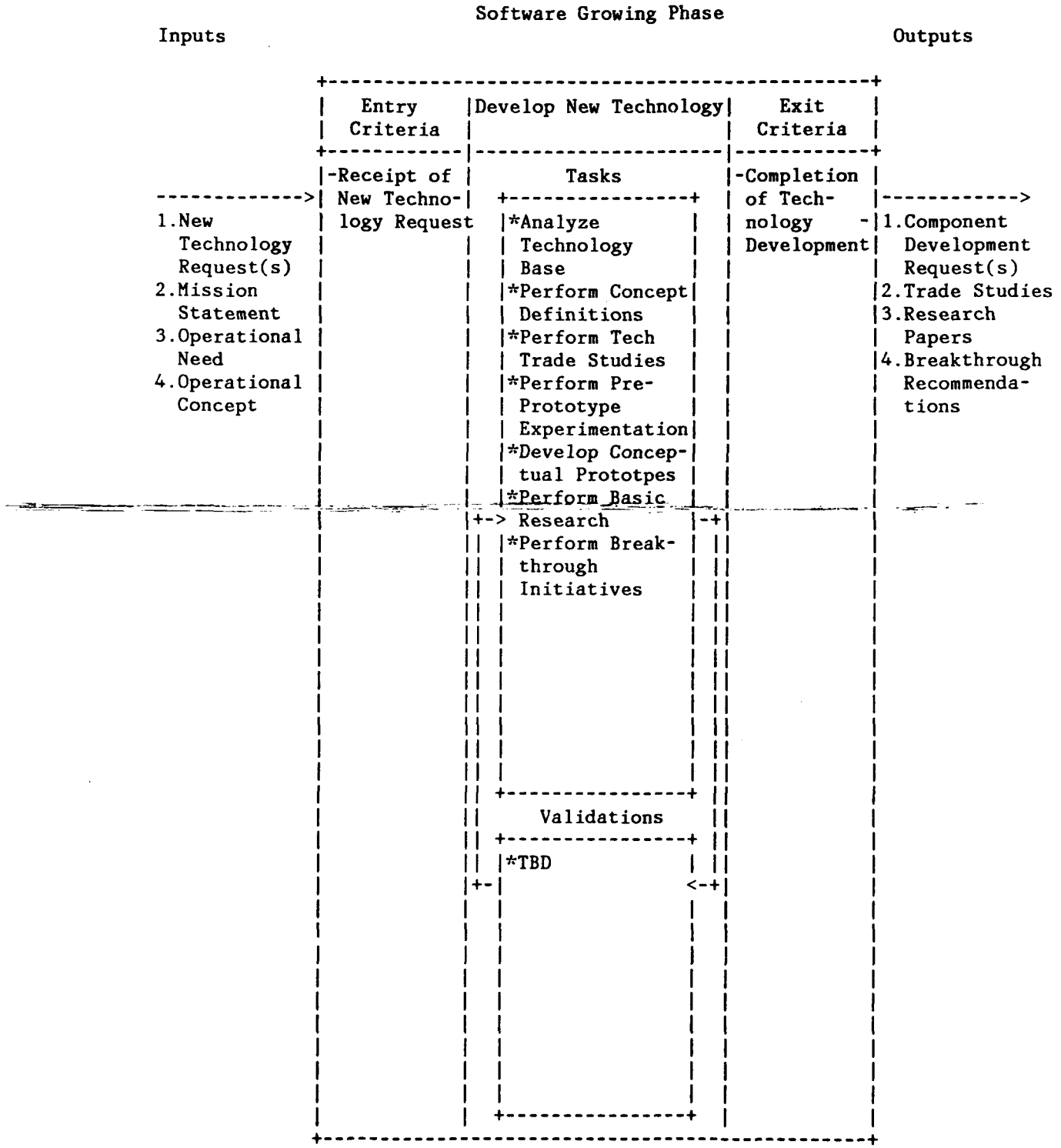


Figure 20. Develop New Tecnology Process Model

Productization and Production Process Model

The process model for the Productization and Production Phase is shown in Figure 22 on page 49.

The entry criteria for this phase is the succesful completion of the Pre-Productization Review. This includes approval of all inputs by the customer team. The inputs, as shown in the fugure, are created by the System Architecture Phase.

The tasks performed during the Productization and Production Phase are:

- Productize Full-Capability System Prototype
- Test Productized System
- Run Acceptance Test of System and
- Deliver System(s).

The relationship between Productization and Production tasks is shown in the figure below.

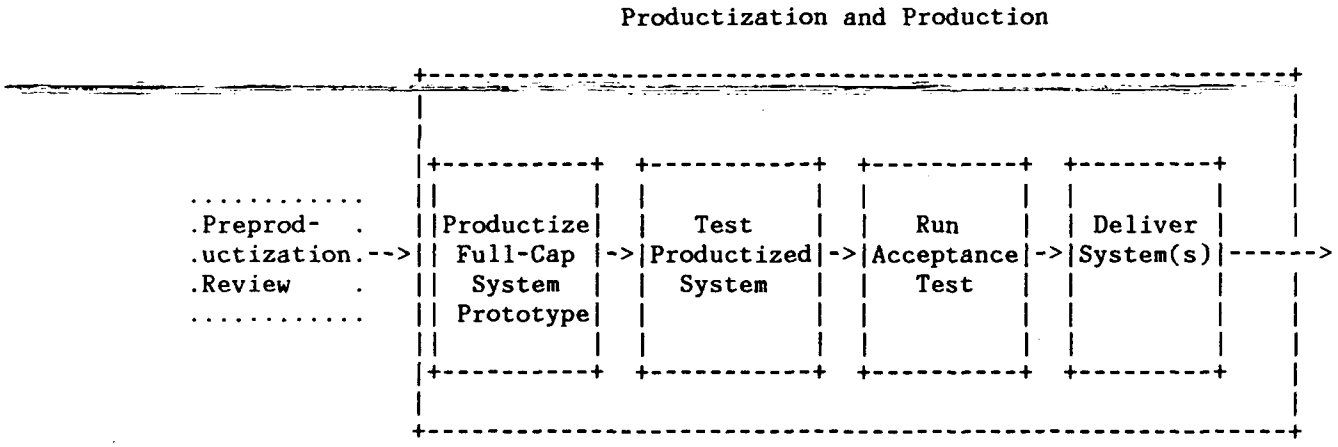


Figure 21. Productization and Production Tasks

The Productize Full-Capability System Prototype task converts the full-capability system prototype into a fully operational (productized) system. Existing reusable components are adapted and new components are developed as required. The Test Productized System task verifies the existence and integrity of the operational capabilities. The Run Acceptance Test of the System task demonstrates the system to the customer team for delivery approval. The Deliver System(s) task deploys systems at each of the customer sites.

The validations performed within the Productization and Production phase include a successful internal review of the system operation and support procedures, a successful system test, and customer acceptance of the operational system. The internal review of the system operation and support procedures, performed during the Productize Full-Capability System Prototype task, ensures that the defined procedures are sufficient to operate and support the system. The system test, performed during the Test System task, verifies the existence and integrity of the operational system capabilities. Customer acceptance of the operational system, occurs as the result of successful completion of the Run Acceptance Test of the System task.

The exit criteria for the Productization and Production Phase is the successful delivery of the productized system to the customer site(s). The outputs of this task include the operational system and its operation and support procedures, and the definitized system design.

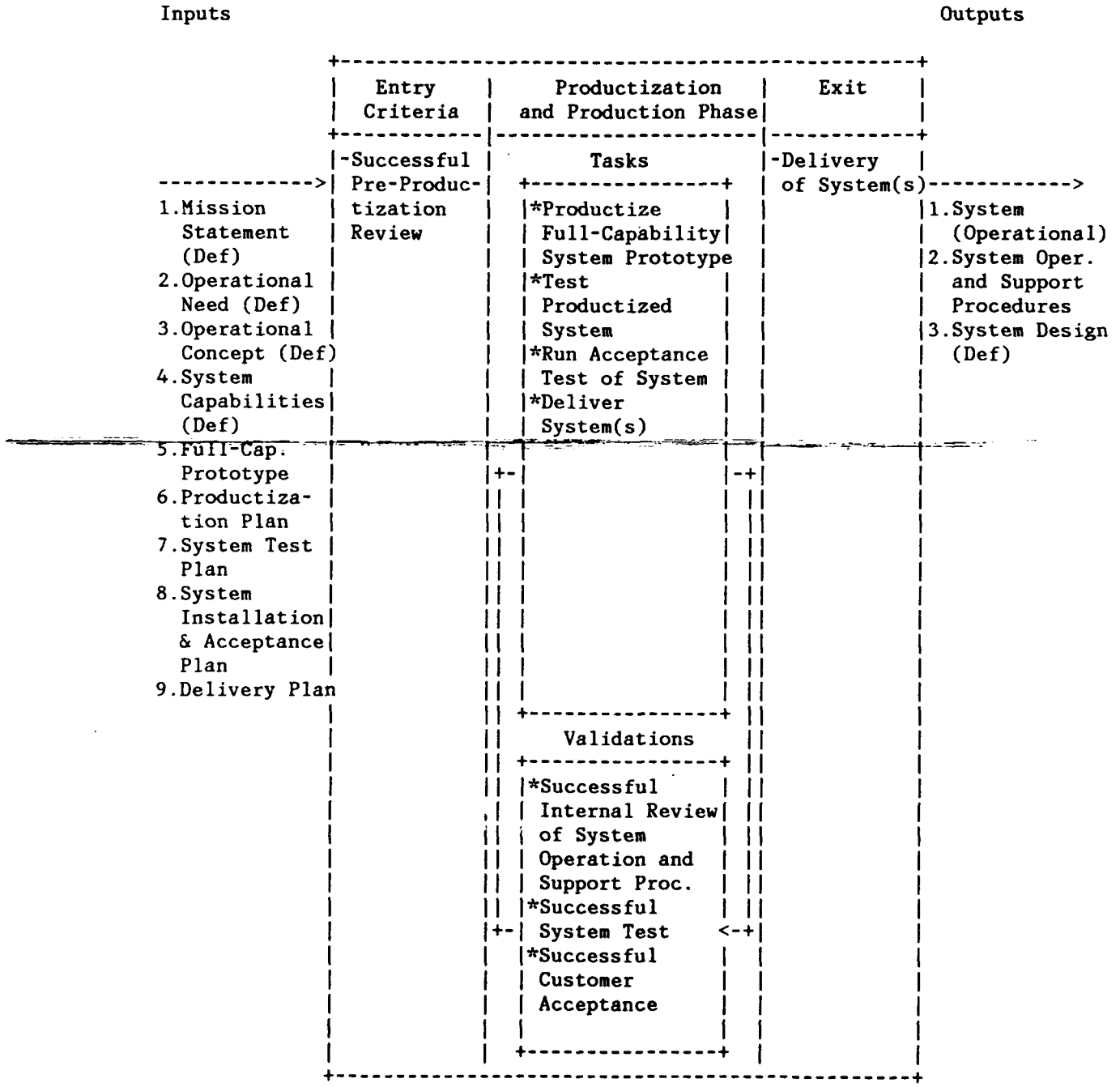


Figure 22. Productization and Production Process Model

System Operation and Support Process Model

The process model for the System Operation and Support Phase is shown in Figure 24 on page 51.

The entry criteria for this phase is the successful delivery of the system at the customer site(s). The inputs, as shown in the figure, are created during the Productization and Production Phase.

The tasks performed during the System Operation and Support Phase are:

- Operate System
- Maintain System and
- Identify New Operational Capabilites.

The relationship between System Operation and Support tasks are shown in the figure below.

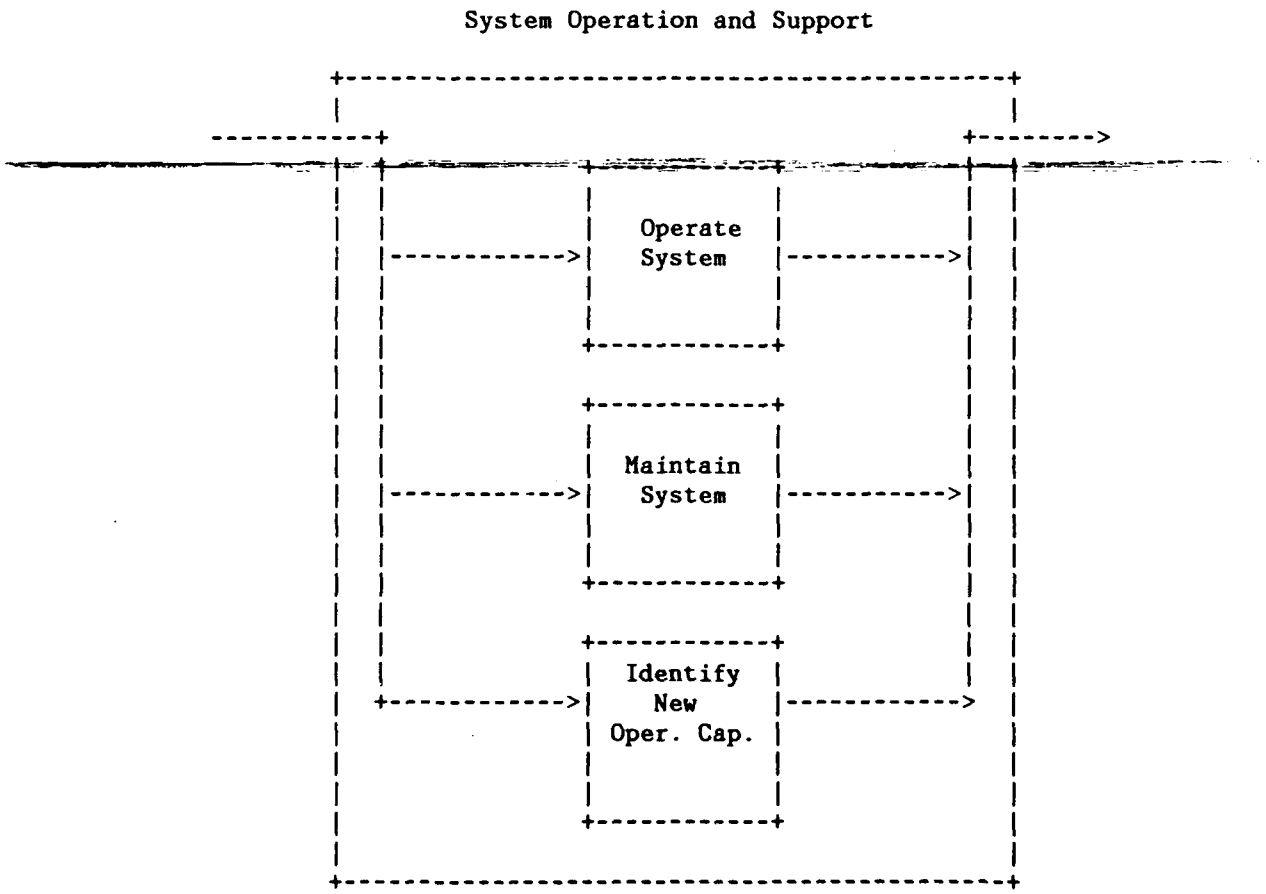


Figure 23. System Operation and Support Tasks

The Operate System task includes the operation and use of the system at customer sites. The Maintain System task provides support for the operation of system and provides upgrades to system capabilities when necessary. The Identify New Operational Capabilities task periodically analyzes the existing system and makes recommendations for new operational capabilities.

The exit criteria for the System Operation and Support Phase is the termination of System Operation.

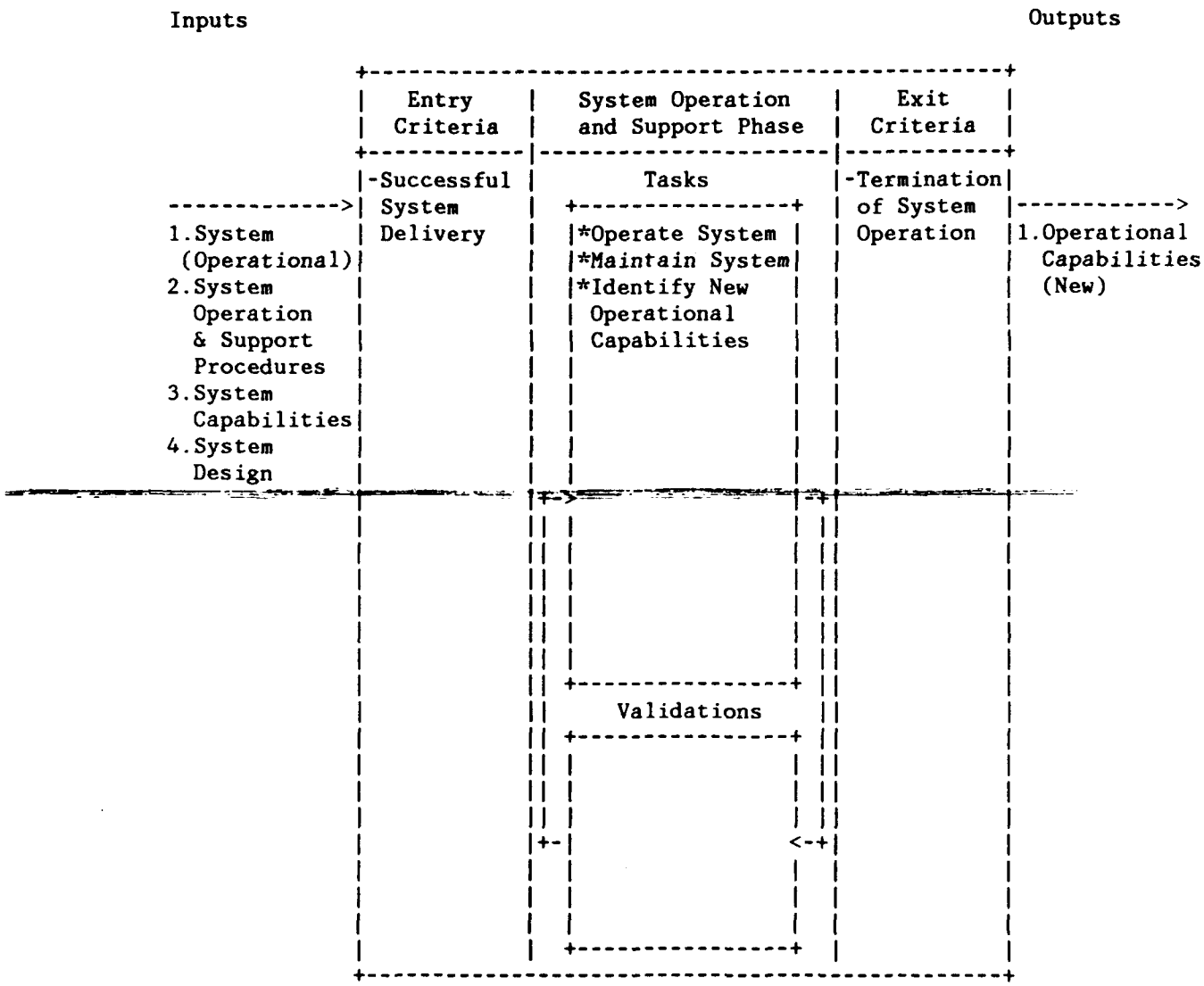


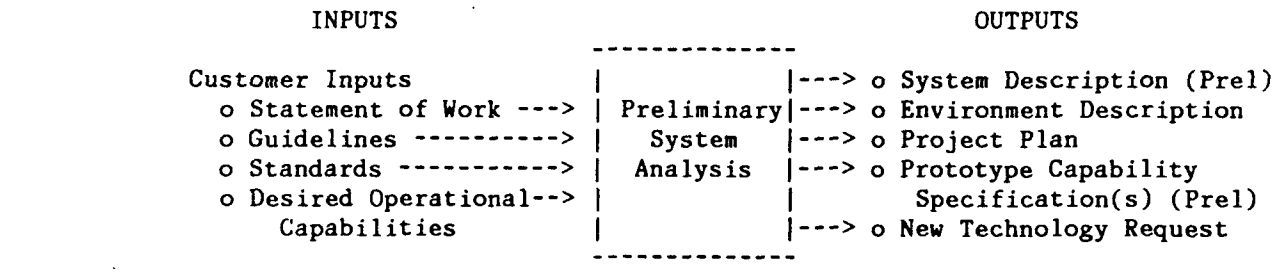
Figure 24. System Operation and Support Process Model

Major Work Products of the SFLC Phases

This section identifies the major input, output, and intermediate work products of each SFLC phase. It describes the role of each work product in the phase, and the context in which it is used.

Note: Some of the work products identified in this section are composed of multiple process model inputs/outputs. The workproducts were defined to encapsulate related information. Regardless of their compostion, the information in the work products corresponds to the process model inputs and outputs.

Preliminary System Analysis



Intermediate Work Products

- o User Interview Summaries
- o Trade Study Reports

The initial phase of the SFLC is Preliminary System Analysis. The purpose of this phase is to perform a preliminary capabilities analysis in order to obtain a high level understanding of users' requirements and system capabilities. The analysis must be at a level which enables the development of a project plan and a preliminary capability specification for the initial system prototype(s). It should also identify new technologies which need to be researched and developed.

The Preliminary System Analysis phase begins with the receipt of the Statement of Work and any guidelines and standards from the customer which need to be followed. The customer may also supply work products that describe required operational capabilities for the system to be developed.

The major output work products of the Preliminary System Analysis phase are the System Description, the Environment Description, the Project Plan, and the Prototype Capability Specification(s) for the initial system prototype(s). These work products provide a description of the system and its operational capabilities and are usually produced in a preliminary form. In ad-

dition, New Technology Requests are produced to initiate efforts to develop new technologies that may be required by the system.

At the start of the Preliminary System Analysis phase, the System Description is developed. It consists of: the Mission Statement, the Operational Need Document and the Operational Concept Document. The Mission Statement is a concise high level description of what the system does. The Operational Need Document describes the operational problem to be solved in user operational terms rather than technical engineering terms. The Operational Concept Document describes the high level features of the system and how the system will operate in the environment in which it will to be deployed. The Mission Statement is completed and signed-off in the System Analysis phase. The other two documents produced are preliminary and are expanded and definitized in the System Architecture phase.

The Project Plan consists of all of the major planning documents (e.g. Incremental Build Plan) necessary for project management. It is incrementally expanded and updated as the project life cycle is executed. During the System Analysis Phase, effort is focused on developing the Incremental Build Plan, Usability Plan, Software Development Plan and the Prototyping Plan(s) for the initial system prototype(s). The Incremental Build Plan defines the planned order in which the system capabilities will be built and integrated. This plan is refined based on the results of prototype evaluations and increased understanding of the required system capabilities. The Usability Plan outlines objectives to ensure that the system's user interface meets the needs of the end user. It takes into consideration any required user interface standards input by the customer and the results of the user interviews conducted during this phase. The Software Development Plan defines the methods to be used during software development. The initial Prototyping Plan defines the approach for building and evaluating the initial system prototype.

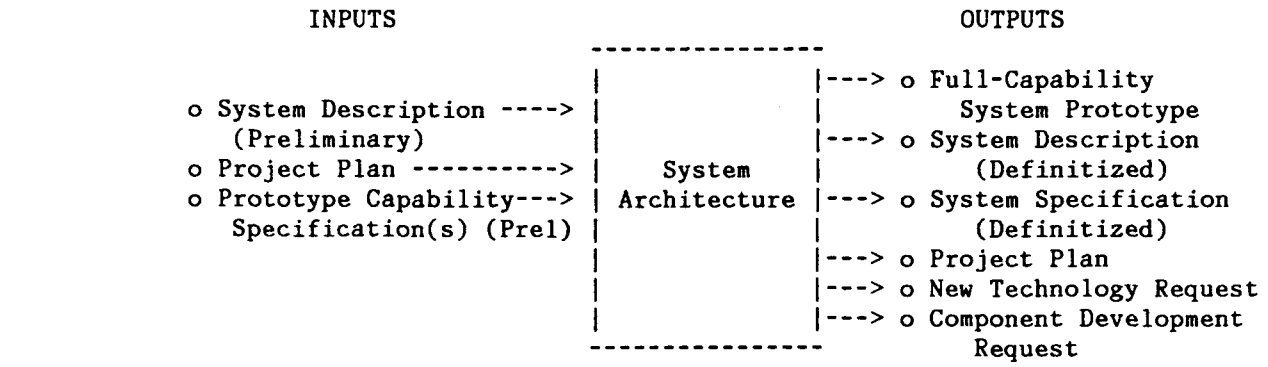
The Environment Description details the environment the contractor will use while performing its life cycle responsibilities. It consists of the specification, identification and instantiation of the Environment. The Environment Specification states the environment capabilities necessary to complete the contractor's life cycle responsibilities. The Environment Identification identifies the actual environment features which will fulfill the environment requirements. Lastly, the chosen hardware and software is integrated into the environment. For example, a project that has to compile Ada source code would record this as a capability specification. An Ada compiler would then be identified and integrated into the system. As the environment is modified the Environment Description should be updated.

The preliminary Prototype Capability Specification defines the capabilities to be developed in the initial system prototype(s). Included in the specification is the identification of the reusable software components to be used and the specification of new components to be developed. (The results of studies concerning trade-off decisions are recorded in Trade Study Reports.) The Prototype Capability Specification in conjunction with the Prototyping Plan (part of the Project Plan) drives the first iteration of the System Architecture phase.

Once the above work products are developed the Pre-Prototyping Review takes place. The Pre-Prototyping Review is performed to ensure that the system developer has a sufficient understanding of the users' requirements and system operational capabilities to build a prototype. This review also ensures an understanding between the customer team and the system developer of the life cycle approach, milestones and deliverables to be followed by the project.

Note: A System Specification (Preliminary) may be input by the customer, but the SFLC only requires a definitized System Specification after a full-capability prototype has been developed.

System Architecture



Intermediate Work Products

- o System Prototypes
- o System Description (Expanded)
- o Prototype Capability Specifications
- o Prototype Designs
- o Project Plan
- o Trade Study Reports (e.g., HW/SW)
- o System Operation and Support Documents

The System Architecture Phase is the driving phase of the SFLC. Key goals of this phase are to formulate and definitize the system capabilities that satisfy the requirements of the customer/user, and reduce technological and programmatic risks in the development of the operational system. Instrumental in achieving these goals is the evolution of a full-capability prototype of the system with maximum use of reusable components. As part of the process of evolving a full-capability system prototype, incremental system prototypes are developed concurrently with new software components and new technologies. Incremental prototypes addressing alternative approaches of the same capability, or addressing distinct capabilities are also developed in parallel. These prototypes address technological and programmatic risks early in the system life cycle and allow early, effective customer involvement in evaluating and refining the system capabilities.

The major input work products to the System Architecture phase come from the System Analysis phase: the preliminary System Description, the Project Plan and the preliminary Prototype Capability Specification(s) for the initial system prototype(s). The major output work products of this phase include the full-capability System Prototype, the Project Plan and the definitized System Description and System Specification.

Through successive iteration, the initial system prototype is evolved to a full-capability system prototype. The incremental building of the full-capability requires several key intermediate work products including

- the Project Plan and Prototype Capability Specifications which drive each iteration,

- the interim capability prototypes and their designs which are the result of each iteration,

- Trade Study Reports which document the results of the analysis of alternative approaches within an iteration, or across iterations.

In addition, requests can be made to the Software Growing phase for New Technology and Component Development.

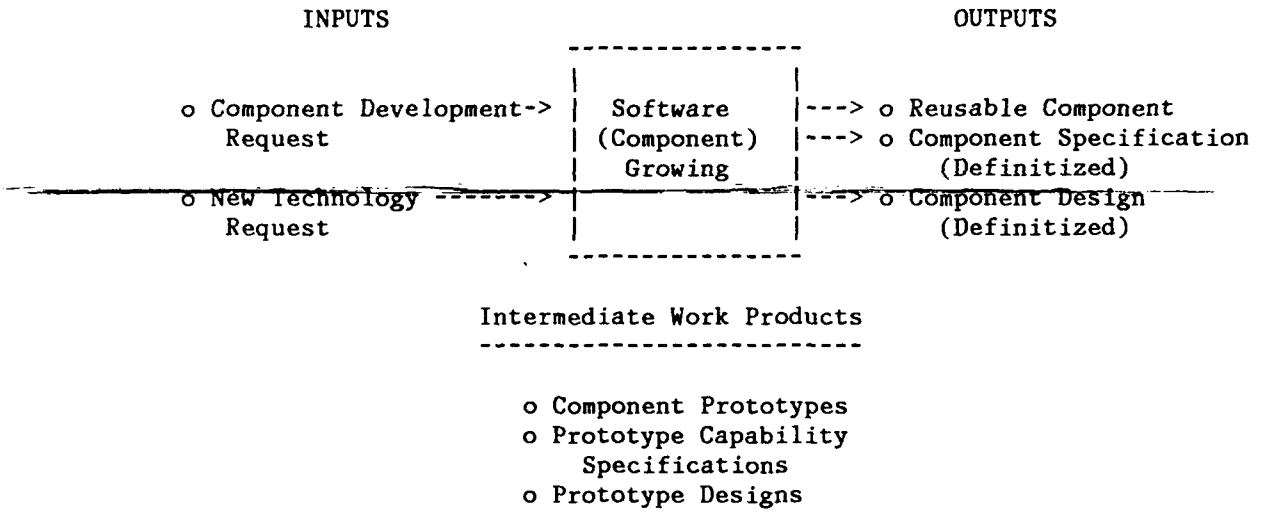
The Prototype Capability Specifications for the system, in conjunction with the Incremental Build Plan and the Prototyping Plan (parts of the Project plan) drive each iteration in the System Architecture phase. As described in the System Analysis phase, the Incremental Build Plan defines the planned order in which system capabilities will be built. A Prototyping Plan, which defines the approach for building and evaluating a prototype, is developed for each prototype built during an iteration. The Prototype Capability Specification defines the capabilities which will exist in the prototype at the end of an iteration.

The interim System Prototypes are key work products within this phase. These prototypes are the mechanism the customer/user can effectively use to evaluate and refine system capabilities. The evaluation results are captured in the Prototype Capability Specification. They influence decisions concerning succeeding iterations. The interim System Prototypes allow trade-off studies (e.g. Software/Hardware, Performance/Space) to be performed and recorded in Trade Study Reports. The system prototypes help refine the system description and enable a high level architecture to be definitized and recorded in the System Description Document. They also enable the exploration of risks as discussed previously.

At the end of an iteration a decision is made whether or not a full-capability System Prototype exists. If one does exist, the System Specification, which defines the system capabilities that will be included in the productized system, is definitized. This document, in conjunction with the full-capability System Prototype fully describe the system to be productized and the Productization Phase begins. If a full-capability System Prototype does not exist, then the next iteration's Prototype Plans, and Prototype Capability Specifications are developed.

Note: The System Operation and Support documents are those documents required to support the operational system such as a system user manual or an installation manual. The System Operation and Support documents to be delivered for a system will vary from system to system. During this phase, trade-off analyses are done to determine which System Operation and Support documents will be developed, and the content of each document. For example, a system may not require an extensive user manual if an informative help/tutorial component is a part of the system. These documents are fully developed during the Productization and Production phase.

Software Growing

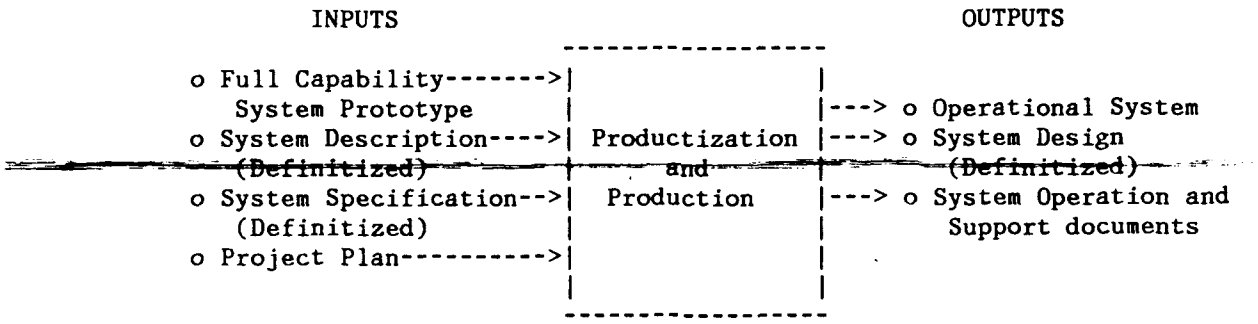


During the Software Growing phase, new software components are developed and new technologies are explored. The prototyping cycle for developing new software components, performed concurrently and iteratively, is similar to the System Architecture phase except emphasis is placed on building reusable component for the repository. Since the development cycle is similar, so are many of the major work products. The major difference is that the work products of this phase are scoped for the component level and not the system level.

The major input work products of the Software Growing phase are requests from other phases for the development of new components and/or new technologies. The major output work products are the resulting Reusable Components and their definitized Specification and Design. The intermediate work products include the Prototype Capability Specification and Design for the component and the interim Component Prototypes.

Note: Work products for developing new technologies have not been addressed since the process has not been defined at this time.

Productization and Production



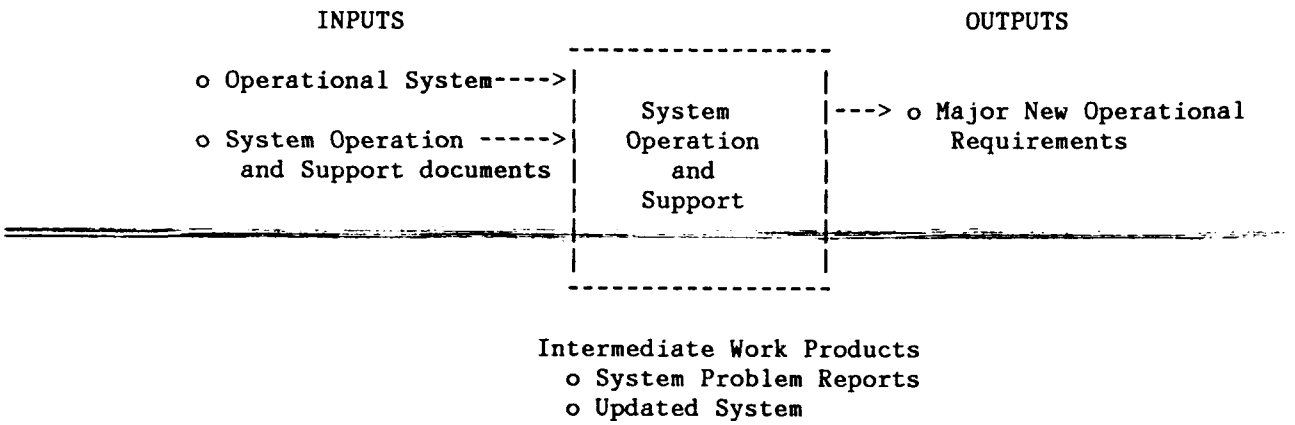
In the Productization and Production phase, the final system prototype is productized and delivered to the customer. The major input work products are the full-capability System Prototype, the definitized System Description and System Specification and the Project Plan, which were developed in the System Architecture phase. The major output work product of this phase is the productized system: "a fully defined, specified, developed, documented, tested, delivered, and supported production quality software/hardware system (STAR87)". Supporting output work products include the definitized System Design and the System Operation and Support documents.

The key areas of the Project Plan during this phase are the Productization Plan, the System Test Plan, the System Installation and Acceptance Plan and the Delivery plan. The Productization Plan defines the steps necessary to productize the full-capability system prototype. The System Test Plan defines the testing objectives for the system. The System Installation and Acceptance Plan and the Delivery Plan define the steps necessary to deliver and install the system at customer sites.

The definitized System Design is developed during this phase to baseline the design of the system which will be deployed. Most of the information necessary for completion of this document can be gathered from the prototype system designs and the component designs previously developed.

During this phase, the System Operation and Support documents identified and outlined in the System Architecture phase are fully developed and definitized.

System Operation and Support



In the System Operation and Support phase, the productized system is deployed and operated in its intended environment. It may be maintained by the customer, the system development contractor, or a maintenance contractor. As the system is operated, system problems may be identified. If so, a solution is developed and the system is updated. If new operational requirements are identified for the next version of the system, these are input to the Systems Analysis phase. The SFLC is then repeated, as required, for a new version of the operational system.

Summary of Work Products

This section presents a summary description of the work products in table form. The first table describes the major inputs and outputs of each phase of the SFLC. The second table describes the development of each major work product during the execution of the SFLC. The third table lists the external definitized work products that describe the system. The last table lists the external definitized work products that describe the reusable components generated during the execution of the SFLC.

SFLC Phase Inputs and Outputs

This table describes the major inputs and outputs of each phase of the SFLC.

Phase	Inputs	Outputs
Preliminary System Analysis	Statement of Work, Guidelines, Standards, Required Operational Capabilities	System Description (Prel), Environment Description, Project Plan, Prototype Capability Specifications (Prel), New Technology Request
System Architecture	System Description (Prel), Project Plan, Prototype Capability Specifications (Prel)	Full-Capability System Prototype, System Description (Def), System Specification (Def), Project Plan
Software Growing	Software Component Request, New Technology Request	Component, Component Specification (Def), Component Design (Def),
Productization and Production	Full-Capability System Prototype, System Description (Def), System Specification (Def), Project Plan	Operational System, System Operation and Support Documents, System Design (Def)
System Operation and Support	Operational System, System Operation and Support Documents	Major New Operational Requirements

Work Product Development Stages

This table describes the development of each of the major work products during the execution of the SFLC.

Work Product	Preliminary System Analysis	System Architecture	Software Growing	Productization and Production
Environment Description	Preliminary	Updated	Updated	Updated
Project Plan	Preliminary	Updated	Updated	Updated
System Description	Preliminary	Definitized		
Trade Study Reports	Preliminary and Definitized	Definitized		
Prototype Capability Spec	Preliminary (Initial System Prototype)	Expanded (System Prototype)	Preliminary and Expanded (Component Prototype)	
Prototype Design		Preliminary and Expanded	Preliminary and Expanded	
System		Prototyped		Productized
System Specification		Definitized		
System Operation and Support Documents		Outlined		Definitized
System Design				Definitized
Component			Prototyped and Definitized	
Component Design			Definitized	
Component Specification			Definitized	

Definitized External Work Products for the System

At the completion of the life cycle, the definitized external work products describing the system include:

Name	Type	Output Phase
System Description	Document	System Architecture
System Specification	Document	System Architecture
System Design	Document	Productization and Production
System Operation and Support Documents	Document	Productization and Production
Operational System	System	Productization and Production

Definitized External Work Products for Components

At the completion of the life cycle, the definitized external work products describing components include:

Name	Type	Output Phase
Component Specification	Document	Software Growing
Component Design	Document	Software Growing
Component	Software	Software Growing

Acronyms

Acronym	*Meaning*
CDRL	Contract Data Requirements List
DoD	(United States) Department of Defense
IBM	International Business Machines
SFLC	Software-First Life Cycle
SOW	Statement of Work
STARS	Software Technology for Adaptable, Reliable Systems

References

(Char88) The Charles Stark Draper Laboratory, Inc., Organizer, *Final Report - DARPA Concurrent Design/Concurrent Engineering Workshop*, December 1988.

(Gree89) Greene, Joseph S. Jr., Colonel, USAF, "Position Paper - Subject: Software-First", Defense Advanced Research Projects Agency, 1989.

(Hump89) Humphrey, Watts S., *Managing the Software Process*, Addison-Wesley Publishing Co., 1989.

(IBM1380) IBM Systems Integration Division, *Consolidated Reusability Guidelines*, CDRL Sequence No. 0380, March 1989.

(IBM1490) IBM Systems Integration Division, *C3 Application Blueprint*, CDRL Sequence No. 1490, Jan 1990.

(IBM1540) IBM Systems Integration Division, *Repository Guidebook(Draft)*, CDRL Sequence No. 1540, August 1989.

(Luqi89) Luqi, "Software Evolution Through Rapid Prototyping", *IEEE Computer* , May 1989.

(Prie87) Prieto-Diaz, Ruben, "Domain Analysis for Reusability", *COMPSAC 87, The Eleventh Annual International Computer Software and Applications Conference*, 1987.

(Radi85) Radice, R.A., Roth, N.K., O'Hara,Jr., A.C. and Ciarfella, W.A., "A Programming Process Architecture", *IBM Systems Journal*, Vol 24, No. 2, 1985

(Royc70) Royce, Winston W., "Managing the Development of Large Software Systems", *Proceedings, IEEE WESCON*, August 1970.

(STAR87) STARS Joint Program Office, Greene, Joseph S. Jr., Colonel, STARS Program Manager, "Software-First Systems Development Standard For Systems-In-The-Large", *Attachment A.5, STARS SOW Materials*, 1987.

Appendix A. SFLC Work Product Descriptions

Definitions of the major work products are listed alphabetically in this section. The template used to define the work products includes the name of the work product, its type, purpose, and content, how it is developed, and whether it is an external or internal work product. Task IR66, Software-First CDRLS, will refine these definitions, and define each work product's format.

Name:
Component

Type:
Software.

Description:

A reusable, self-contained software portion of the system.

Content:

- Documented Software

Format:
TBD

Development:

A Component Development Request, input to the Software Growing phase, initiates the development of a software component. The component is prototyped and productized during the Software Growing phase.

External/Internal:

The productized component is an external work product. The interim prototypes may be internal or external.

Figure 25. Component

Name:
Component Design

Type:
Document.

Description:

The definitized Component Design details the implementation of a Component.

Content:

- Architecture
- Interfaces

Format:
TBD

Development:

There is a definitized Component Design document for each Component developed in the Software Growing phase.

External/Internal:

The definitized Component Design is an external work product.

Figure 26. Component Design

Name:
Component Specification

Type:
Document.

Description:

The definitized Component Specification details the capabilities of a Component.

Content:

- Capabilities

Format:
TBD

Development:

There is a definitized Component Specification for each Component developed in the Software Growing phase.

External/Internal:

The definitized Component Specification is an external work product.

Figure 27. Component Specification

Name:
Environment Description

Type:
Document, Software/Hardware.

Description:

This document details the environment the contractor will use while performing its life cycle responsibilities.

Content:

- Environment Specification - The statement of the environment requirements necessary to complete the life cycle responsibilities. (e.g. Compile Ada Source Code)
- Environment Identification - The statement of the actual environment features selected to fulfill the environment requirements. (e.g. Ada Compiler)
- Environment Instantiation - The actual HW/SW of the identified environment features.

Format:
TBD

Development and Use:

The Environment Description should cover all phases of the life cycle. It is drafted in the Preliminary System Analysis phase and periodically updated during the other phases.

External/Internal:

The Environment Description is an external work product.

Figure 28. Environment Description

Name:
New Technology Request

Type:
Document.

Description:

Formal description of a problem where breakthrough solutions are desired.

Content:
TBD

Format:
TBD

Development and Use:

New Technology Requests can be initiated from any phase, but most requests will be initiated from the Preliminary System Analysis, System Architecture and Software Growing phases.

External/Internal:

New Technology Requests can be internal or external work products.

Figure 29. New Technology Request

Name:
Project Plan

Type:
Document.

Description:

The Project Plan consists of all of the major planning documents necessary for project completion.

Content:

- Incremental Build Plan
- System Engineering Plan
- Software Development Plan
- Hardware Development Plan
- Prototyping Plans
- Usability Plan
- Productization Plan
- System Test Plan
- System Installation and Acceptance Plan
- Delivery Plan

Format:
TBD

Development:

This document is incrementally expanded and updated as the project life cycle is executed.

External/Internal:

The Project Plan is a working document and at different points in the life cycle a snapshot of it is externalized.

Figure 30. Project Plan

Name:
Prototype Capability Specification

Type:
Document.

Description:

The Prototype Capability Specification details the capabilities of the System or Component Prototype to be developed.

Content:

- Previous Capabilities
- New Capabilities
- Identification of Reusable Components
- Identification of standard interfaces

Format:
TBD

Development:

A Prototype Capability Specification is developed for each prototype under development. This document in conjunction with the Prototyping Plan drives the refinement of prototyped capabilities.

External/Internal:

Only the preliminary Prototype Capability Specification for the initial System Prototype is an external work product. All others are informal internal work products.

Figure 31. Prototype Capability Specification

Name:
Prototype Design

Type:
Document.

Description:

The Prototype Design details the implementation of the System or Component Prototype to be developed. (It may include the design of lower level reusable components which are part of this component.)

Content:

- Architecture
- Interfaces

Format:
TBD

Development:

A Prototype Design is developed for each prototype under development.

External/Internal:

All Prototype Designs are informal, internal work products.

Figure 32. Prototype Design

Name:
Software Component Development Request

Type:
Document

Description:

This document formally describes the capabilities of a software component which must be developed.

Content:
TBD

Format:
TBD

Development and Use:

Software Component Development Request can be initiated from any phase, but most requests will be initiated from the System Architecture phase.

External/Internal:

Software Component Development Requests can be internal or external work products.

Figure 33. Software Component Development Request

Name:
System

Type:
Software

Description:

The software system to be developed.

Content:

- Documented Software

Format:
TBD

Development:

During the Software Architecture phase, incremental prototypes are developed until a full-capability system prototype exists. This full-capability prototype is productized during the Productization and Production phase resulting in the Operational System.

External/Internal:

The productized system is an external work product. The interim and full-capability prototypes are usually internal work products, however some may requested to be external.

Figure 34. System

Name:
System Description

Type:
Document.

Description:

The System Description contains a high level description of the system mission, the operational need and operational concept.

Content:

- Mission Statement
- Operational Need
- Operational Concept

Format:
TBD

Development:

See: Mission Statement, Operational Need Document and Operational Concept Document.

External/Internal:

The definitized System Description is an external delivery.

Figure 35. System Description

Name:
System Description - Mission Statement

Type:
Document.

Description:

The Mission Statement is a concise high level description of what the system does. It relates the mission of the overall system to the desired capabilities of the operational system.

Content:

- Purpose of the system
- Goals of the system

Format:
TBD

Development:

The Mission Statement is developed during the Preliminary System Analysis phase.

External/Internal:

The definitized System Description, including the Mission Statement is an external delivery.

Figure 36. System Description - Mission Statement

Name:
System Description - Operational Concept

Type:
Document.

Description:

The Operational Concept Document describes the high level features of the system and how the system will operate in the environment in which it will be deployed.

Content:

- Operational Environment Description
- System Functional Overview
- High Level System Architecture

Format:
TBD

Development:

A preliminary version of the System Operational Concept document is drafted in the Preliminary System Analysis phase. It is definitized during the System Architecture phase.

External/Internal:

The definitized System Description, including the Operational Concept is an external delivery.

Figure 37. System Description - Operational Concept

Name:
System Description - Operational Need

Type:
Document.

States:
External.

Description:

The Operational Need Document describes the operational problem to be solved in user operational terms rather than technical engineering terms.

Content:

- Identification and evaluation of existing system deficiencies
- Identification of additional operational capabilities
- Prioritization of capabilities
- Identification of constraints

Format:
TBD

Development:

A preliminary version of the Operational Need document is developed during the Preliminary System Analysis Phase. It is definitized during the System Architecture Phase.

External/Internal:

The definitized System Description, including the Operational Need is an external delivery.

Figure 38. System Description - Operational Need

Name:
System Design

Type:
Document.

Description:

The definitized System Design details the implementation of the operational system.

Content:

- System Architecture
- Interfaces

Format:
TBD

Development:

The System Design is definitized during the Productization and Production phase.

External/Internal:

The definitized System Design is an external document.

Figure 39. System Design

Name:
System Specification

Type:
Document.

Description:

The definitized System Specification defines the system capabilities that satisfy the customer/user needs and will be included in the productized system.

Content:

- Capabilities

Format:
TBD

Development and Use:

The definitized System Specification is a major output of the System Architecture phase.

External/Internal:

The definitized System Specification is an external document.

Figure 40. System Specification

Name:
System Operation and Support Documents

Type:
Document.

Description:

The collection of documents necessary for system operation and support. Examples include a System User's Manual and Installation Manual.

Content:
TBD

Format:
TBD

External/Internal:

The System Operation and Support Documents selected are external work products.

Figure 41. System Operation and Support Documents

Name:
Trade Study Report

Type:
Document.

Description:

A Trade Study Report documents the results of the analysis of alternative approaches, such as hardware/software trade-offs, and performance/space trade-offs.

Content:

- Experiment
- Results

Format:
TBD

Development and Use:

Trade Studies are usually performed in the System Architecture and Software Growing phases.

External/Internal:

Trade Study Reports can be internal or external work products.

Figure 42. Trade Study Report

Appendix B. Major Site Visits and Meetings

The following site visits and meetings have been conducted to provide a forum for presenting, and receiving comments, on the SFLC definition and to gain insight and feedback from SFLC related industry experience.

- Ben Flores, IBM SID Houston, Space Shuttle Project - June 8, 1989

  The purpose of this meeting was to discuss the successful use of a prototype life cycle by the PCASS project and the major lessons learned.

- Dr. Winston Royce, President of Software First, Inc.- August 1, 1989

  The purpose of this meeting was to discuss the Leonardo Project, which is being developed for the Micro-Computer Consortium. Leonardo is an effort to implement a framework and environment for developing software systems. Dr. Royce also presented his views on the SFLC.

- IBM SID User Interface Center of Competency - August 3, 1989

  The purpose of this meeting was to present the the SFLC to the User Interface Center of Competency and exchange ideas on the role and impact of prototyping on the system development life cycle.

- Elliot Margolis, IBM SID, FAA/AAS Project - September 8, 1989; Dr.Russ Benel, IBM SID, FAA/AAS Project - October 6, 1989

  The purpose of this meeting was to discuss the FAA Advanced Automatation System (AAS) project's use of prototyping in their life cycle and the lessons learned from it.

- EIA Computer Resources Workshop, Panel 3D, "DOD-HDBK-287 A Tailoring Guide for DOD-STD-2167 - Review Recommendations" - September 18, 1989

  The purpose of this meeting was to present the SFLC to the panel in order to gain their feedback and insights, and to provide an example of a new and emerging development process with which to measure the flexibility of their handbook.

- IBM SID Gaithersburg Software Engineering Council - September 27, 1989

  The purpose of this meeting was to present the SFLC to the IBM Gaithersburg Software Engineering Council, in order to gain their feedback and insights, based on their own broad experiences in developing large software systems.

- Ed Seidewitz and Frank McGarry, Goddard Space Flight Center - October 17, 1989

  The purpose of this meeting was to discuss their efforts over the past few years in developing and reusing Ada components, using object oriented design, and their current effort to define and implement a generic reusable model for a specific application domain.

- Elfrieda Harris, Goddard Space Flight Center - October 17, 1989

  The purpose of this meeting was a demonstration of Transportable Application Environment (TAE), a portable UNIX based prototyping system that includes an Ada code generator. TAE Plus Version 4.0 will be installed and evaluated for use on STARS by the IBM Gaithersburg User interface Center of Competency.

- Dave Amos, IBM Toronto, Application Development Lab - October 18, 1989

  The purpose of this meeting was to discuss their methodology, used over the past several years, for developing reusable components. Their methodology involves the use of a standardized and reusable framework and environment; and includes the use of Excelerator to analyze, design, assemble, and execute systems from reusable components.

Appendix C. Bibliography

1. Boehm, Barry W., "Understanding and Controlling Software Costs", *IEEE Transactions on Software Engineering*, Vol. 14, October 1988.

2. Boehm, Barry W., "A Spiral Model of Software Development and Enhancement", *IEEE Software*, May 1988.

3. Boehm, Barry W., Gray, Terrence E. and Seewaldt, Thomas, "Prototyping Versus Specifying: A Multiproject Experiment", *IEEE Transactions on Software Engineering*, Vol SE-10, May 1984.

4. Boeing, "General Sofware Development Plan", *STARS CDRL 00670*, 1989.

5. Carr, Marvin J., "A Circular Model for Software Development", *Proceedings of the Sixth Washington Ada Symposium*, June 1989.

6. The Charles Stark Draper Laboratory, Inc., Organizer, *Final Report - DARPA Concurrent Design/Concurrent Engineering Workshop*, December 1988.

7. Connel, John L. and Shafer, Linda Brice, *Structured Rapid Prototyping*, Yourdon Press, 1989.

8. Cooper, Jack, "Software Development Management Planning", *IEEE Transactions on Software Engineering*, Vol. SE-10, January 1984.

9. Davis, Alan M., "A Taxonomy for the Early Stages of the Software Development Life Cycle", *The Journal of Systems and Software*, Vol. 8, 1988.

10. Davis, Alan M., Bersoff, Edward H. and Comer, Edward R., "A Strategy for Comparing Alternative Software Development Life Cycle Models", *IEEE Transactions on Software Engineering*, Vol. 14, No. 10, October 1988.

11. Government Computer News (Interview), "The Men Behind DOD Std 2167 Discuss Its Goals", *Government Computer News*, June 10, 1988.

12. Hamilton, M., and Zeldin, S., "The Functional Life Cycle Model and its Automation: USE.IT", *The Journal of Systems and Software*, Vol. 3, 1983.

13. Humphrey, Watts S.,"Characterizing the Software Process: A Maturity Framework", *IEEE Software*, March 1988.

14. Humphrey, Watts S., *Managing the Software Process*, Addison-Wesley Publishing Co., 1989.

15. IBM Systems Integration Division, *Consolidated Reusability Guidelines*, CDRL Sequence No. 0380, March 1989.

16. IBM Systems Integration Division, *Environment Capability Matrix*, CDRL Sequence No. 0110, March 1989.

17. IBM Systems Integration Division, *Repository Guidebook(Draft)*, CDRL Sequence No. 1540, August 1989.

18. IBM Systems Integration Division, *Software-First Life Cycle - Preliminary Definition*, CDRL Sequence No. 1230, August 1989.

19. Luqi, "Software Evolution Through Rapid Prototyping", *IEEE Computer* , May 1989.

20. Prieto-Diaz, Ruben, "Domain Analysis for Reusability", *COMPSAC 87, The Eleventh Annual International Computer Software and Applications Conference*, 1987.

21. Radice, R.A., Roth, N.K., O'Hara,Jr., A.C. and Ciarfella, W.A., "A Programming Process Architecture", *IBM Systems Journal*, Vol 24, No. 2, 1985

22. Royce, Winston W., "Managing the Development of Large Software Systems", *Proceedings, IEEE WESCON*, August 1970.

23. Scacchi, Walt, "Managing Sofware Engineering Projects: A Social Analysis", *IEEE Transactions on Software Engineering*, Vol. SE-10, January 1984.

24. "Synthesis and DoD-Std-2167", *The Software Productivity Consortium Quarterly*, Vol. 3, No. 2., Spring 1989.

25. STARS Joint Program Office, Greene, Joseph S. Jr., Colonel, STARS Program Manager, "Contract Data Requirements List (DD Form 1423", *Attachment A.7, STARS SOW Materials*, 1987.

26. STARS Joint Program Office, Greene, Joseph S. Jr., Colonel, STARS Program Manager, "Position Paper - Subject: Software-First", Defense Advanced Research Projects Agency, 1989.

27. STARS Joint Program Office, Greene, Joseph S. Jr., Colonel, STARS Program Manager, "Software-First Systems Development Standard For Systems-In-The-Large", *Attachment A.5, STARS SOW Materials*, 1987.

28. "Workshop Examines Software Methods", *The Software Productivity Consortium Quarterly* Vol. 3, No. 2., Spring 1989.